PCT/JP99/03507
30.06.99

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

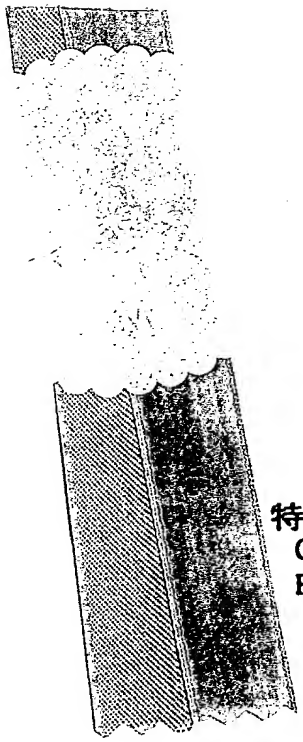
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: １９９８年 ６月３０日

出 願 番 号
Application Number: 平成１０年特許願第１８３１３３号

出 願 人
Applicant (s): アイキュー・ファイナンシャル・システムズ・ジャパン株式会社

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)



１９９９年 ９月２４日

特許庁長官
Commissioner,
Patent Office 近藤隆彦

出証番号 出証特平１１－３０５１６５５

【書類名】　特許願

【整理番号】　BT98001

【提出日】　平成10年 6月30日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 19/00

【発明の名称】　統合金融リスク管理装置および金融取引モデル化装置

【請求項の数】　26

【発明者】

【住所又は居所】　東京都渋谷区鶯谷町１６－４

【氏名】　山崎　裕

【発明者】

【住所又は居所】　千葉県船橋市中野木１－１８－２５　グリーンビレッジ東船橋２０２号

【氏名】　中西　芳明

【発明者】

【住所又は居所】　神奈川県横浜市青葉区荏田西５－１－２８

【氏名】　薄葉　真哉

【発明者】

【住所又は居所】　神奈川県横浜市神奈川区西寺尾３－２８－２４－２０７

【氏名】　山下　司

【発明者】

【住所又は居所】　東京都港区芝浦４－１８－３０

【氏名】　角南　秀幸

【特許出願人】

【住所又は居所】　アメリカ合衆国　ニューヨーク州　ニューヨーク市　リバティー・ストリート　１３０

【氏名又は名称】　バンカース・トラスト・カンパニー（バンカース・トラスト銀行）

【営業所】　東京都千代田区丸の内一丁目３番１号　東京銀行協会

ビルヂング

【代表者】　藤原　久雄

【代理人】

【識別番号】　100106851

【弁理士】

【氏名又は名称】　野村　泰久

【電話番号】　03-3238-0158

【選任した代理人】

【識別番号】　100102336

【弁理士】

【氏名又は名称】　久保田　直樹

【手数料の表示】

【予納台帳番号】　041391

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　統合金融リスク管理装置および金融取引モデル化装置

【特許請求の範囲】

【請求項１】　金融に関する１つ以上の取引実体からなる複合取引に対してリスク管理を行う統合金融リスク管理装置であり、

それぞれ前記取引実体に関する情報を個別に格納する格納手段と、該取引実体に関する時価評価演算手段とを有する１つ以上の取引実体モデル化手段と、

該各取引実体モデル化手段を参照するための参照情報群を保有する参照情報記憶手段と、所定の指示に基づいて、該参照情報群から前記各取引実体モデル化手段を順次参照し、前記時価評価演算手段を実行させてその演算結果を取得し、該各演算結果に基づいて複合取引の特性を算出する複合取引特性算出手段とを有する仮想取引手段と、

を含むことを特徴とする統合金融リスク管理装置。

【請求項２】　請求項１に記載の装置であり、

前記複合取引特性算出手段は、前記各演算結果に所定の変換係数を乗ずる係数乗算手段を含む

ことを特徴とする統合金融リスク管理装置。

【請求項３】　請求項１に記載の装置であり、

前記取引実体手段はリスト構造として実装される、

ことを特徴とする統合金融リスク管理装置。

【請求項４】　請求項１に記載の装置であり、

前記取引実体モデル化手段は、前記格納手段が前記取引実体をモデル化するためのパラメータをクラスメンバとして保持し、前記時価評価演算手段をメソッドとして保持し、オブジェクト指向概念における所定のクラスのインスタンスを実行するモジュールであり、

前記仮想取引手段は、参照情報記憶手段が前記参照情報群をリストメンバとして保持し、前記複合取引特性算出手段を仮想メソッドとして保持し、前記オブジェクト指向概念における所定のコンテナ・クラスのインスタンスを実行するモジュールである、

ことを特徴とする統合金融リスク管理装置。

【請求項５】　所定の取引期間において金融取引対象の受け払いを行うことによって成立する取引系列をモデル化する金融取引モデル化装置であり、

前記金融取引対象の受け側および払い側のそれぞれにおいて、前記所定の取引期間を受けあるいは払い毎に分割して得られる１つ以上の単位取引期間ごとに設けられ、それぞれ、前記単位取引に関する情報を個別に格納する単位取引情報格納手段と、該単位取引期間の時価評価演算手段とを有する１つ以上の単位取引モデル化手段と、

該単位取引モデル化手段を参照するための参照情報群を前記金融取引対象の受け側および払い側のそれぞれに対応して保有する参照情報記憶手段と、所定の指示に基づいて、前記金融取引対象の受け側および払い側のそれぞれにおいて、該参照情報群から前記各単位取引モデル化手段を順次参照し、前記時価評価演算手段を実行させてその演算結果を取得し、該各演算結果に基づいて前記取引系列の特性を算出する取引系列特性算出手段とを有する取引系列モデル化手段と、

を含むことを特徴とする金融取引モデル化装置。

【請求項６】　請求項５に記載の装置であり、

前記所定の期間は所定の時点のみである、

ことを特徴とする金融取引モデル化装置。

【請求項７】　請求項５に記載の装置であり、

前記金融取引対象は通貨以外の金融商品である、

ことを特徴とする金融取引モデル化装置。

【請求項８】　請求項５に記載の装置であり、

前記単位取引モデル化手段はリスト構造として実装される、

ことを特徴とする金融取引モデル化装置。

【請求項９】　請求項５に記載の装置であり、

前記単位取引モデル化手段は、前記単位取引情報格納手段が前記単位取引期間の時価評価演算を行うためのパラメータをクラスメンバとして保持し、前記時価評価演算手段をメソッドとして保持し、オブジェクト指向概念における所定のクラスのインスタンスを実行するモジュールであり、

前記取引系列モデル化手段は、前記金融取引対象の受け側および払い側のそれぞれにおける参照情報群をリストメンバとして保持し、前記取引系列特性算出手段を仮想メソッドとして保持し、前記オブジェクト指向概念における所定のコンテナ・クラスのインスタンスを実行するモジュールである

ことを特徴とする金融取引モデル化装置。

【請求項１０】　請求項５に記載の装置であり、

前記単位取引モデル化手段が有する時価評価演算手段は、該単位取引モデル化手段に対応する前記単位取引期間における、金利、収益率または価格の３種類の指数のいずれかに基づいて計算される将来価値指数と、前記金利、収益率および価格のいずれの指数にも依存しない評価値とに基づいて時価評価演算を実行することを特徴とする金融取引モデル化装置。

【請求項１１】　請求項１０に記載の装置であり、

前記単位取引モデル化手段が有する時価評価演算手段は、その時価評価演算結果に対してさらに所定の割引率を乗じて得られる現在価値をその時価評価演算結果とする割引率乗算手段を含む

ことを特徴とする金融取引モデル化装置。

【請求項１２】　請求項５に記載の装置であり、

日付情報の設定に基づいて、前記所定の取引期間を受けあるいは払い毎に分割して１つ以上の前記単位取引期間を算出し、前記金融取引対象の受け側および払い側のそれぞれにおいて、前記算出した単位取引期間ごとに、前記単位取引モデル化手段を生成し、前記受け側および払い側のそれぞれに対応する前記取引系列モデル化手段内の参照情報群を生成するユーザ・インタフェース手段をさらに含む

ことを特徴とする金融取引モデル化装置。

【請求項１３】　請求項５に記載の装置であり、

前記単位取引モデル化手段ごとに、それがモデル化する金融取引のためのパラメータを変更し、該変更に応じて前記所定の指示を発行するユーザ・インタフェース手段をさらに含む

ことを特徴とする金融取引モデル化装置。

【請求項１４】　金融に関するオプション取引をモデル化する金融取引モデル化装置であり、

それぞれ前記オプション取引の原資産に関する情報を個別に格納する格納手段と、該原資産に関する時価評価演算手段を有する１つ以上の原資産モデル化手段と、

該各原資産モデル化手段を参照するための参照情報群を保有する参照情報記憶手段と、該参照情報群から順次参照される前記各原資産モデル化手段内の前記時価評価演算手段による各演算結果および／または所定の評価モデルに基づいて前記オプション取引の特性を算出するオプション取引特性算出手段とを有するオプションモデル化手段と

を含むことを特徴とする金融取引モデル化装置。

【請求項１５】　請求項１４に記載の装置であり、

前記原資産モデル化手段はリスト構造として実装される、

ことを特徴とする金融取引モデル化装置。

【請求項１６】　請求項１４に記載の装置であり、

前記原資産モデル化手段は、前記格納手段が前記原資産をモデル化するためのパラメータをクラスメンバとして保持し、前記時価評価演算手段をメソッドとして保持し、オブジェクト指向概念における所定のクラスのインスタンスを実行するモジュールであり、

前記オプションモデル化手段は、前記参照情報群をリストメンバとして保持し、前記オプション取引特性算出手段を仮想メソッドとして保持し、前記オブジェクト指向概念における所定のコンテナ・クラスのインスタンスを実行するモジュールである、

ことを特徴とする金融取引モデル化装置。

【請求項１７】　請求項１４に記載の装置であり、

前記オプションモデル化手段は、前記所定の指示の発行時に、前記オプション取引に対して設定されている日付が該オプション取引のイン・ザ・マネーの状態である場合に、前記参照情報群から前記各原資産モデル化手段を順次参照し前記時価評価演算手段を実行させて演算結果を取得し、該各演算結果および前記所定

の評価モデルに基づいて前記オプション取引の特性を算出し、前記日付が前記オプション取引のアウト・オブ・ザ・マネーの状態である場合に、前記所定の評価モデルのみに基づいて前記オプション取引の特性を算出する

ことを特徴とする金融取引モデル化装置。

【請求項１８】　請求項１乃至１７のいずれか１項に記載の装置であり、

前記各時価評価演算機能に対して、該各時価評価演算機能による各時価評価演算時の金融特性を算出し、該金融特性を該各時価評価演算機能に供給する金融特性算出手段をさらに含む、

ことを特徴とする統合金融リスク管理装置または金融取引モデル化装置。

【請求項１９】　請求項１８に記載の装置であり、

前記金融特性算出手段は、前記金融特性をモデル化し、オブジェクト指向概念における所定のクラスのインスタンスを実行するモジュールであり、

前記各時価評価演算機能は、前記所定のクラスのインスタンスが保有し、前記金融特性を算出するための所定のメソッドを参照する参照情報を含む、

ことを特徴とする統合金融リスク管理装置または金融取引モデル化装置。

【請求項２０】　請求項１８に記載の装置であり、

前記金融特性算出手段は、

前記各時価評価演算機能による各時価評価演算時の複数の単一金融特性をそれぞれ算出する複数の単一金融特性算出手段と、

該複数の単一金融特性を合成して新たな仮想金融特性を算出し、該仮想金融特性を前記各時価評価演算機能に供給する仮想金融特性算出手段と、

を含むことを特徴とする統合金融リスク管理装置または金融取引モデル化装置。

【請求項２１】　請求項１８に記載の装置であり、

前記各単一金融特性算出手段は、前記各単一金融特性をモデル化し、オブジェクト指向概念における複数の所定のクラスのそれぞれのインスタンスを実行するモジュールであり、

前記仮想金融特性算出手段は、前記複数の所定のクラスが継承する１つの所定のスーパークラスのインスタンスを実行するモジュールであり、

前記各時価評価演算機能は、前記所定のスーパークラスのインスタンスが保有し、前記仮想金融特性を算出するための所定の仮想メソッドを参照する参照情報を含む、

ことを特徴とする統合金融リスク管理装置または金融取引モデル化装置。

【請求項２２】　請求項１８に記載の装置であり、

前記金融特性は、ネットワークを介して入力されるリアルタイムの金融特性である、

ことを特徴とする統合金融リスク管理装置または金融取引モデル化装置。

【請求項２３】　請求項１８に記載の装置であり、

前記金融特性を定義する金融特性定義手段をさらに含む、

ことを特徴とする統合金融リスク管理装置または金融取引モデル化装置。

【請求項２４】　コンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

それぞれ前記取引実体を個別にモデル化し、それぞれ時価評価演算を実行する１つ以上の取引実体モデル化機能と、

該各取引実体モデル化機能を参照するための参照情報群を保有し、所定の指示に基づいて、該参照情報群から前記各取引実体モデル化機能を順次参照し前記時価評価演算を実行させてその演算結果を取得し、該各演算結果に基づいて前記各取引実体モデル化機能に対応する各取引実体からなる複合取引の特性を算出する仮想取引実現機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。

【請求項２５】　コンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

所定の取引期間において金融取引対象の受け払いを行うことによって成立する取引における該金融取引対象の受け側および払い側のそれぞれにおいて、前記所定の取引期間を受けあるいは払い毎に分割して得られる１つ以上の単位取引期間ごとに実装され、それぞれ該単位取引期間の時価評価演算を実行する１つ以上の単位取引モデル化機能と、

該単位取引モデル化機能を参照するための参照情報群を前記金融取引対象の受け側および払い側のそれぞれに対応して保有し、所定の指示に基づいて、前記金融取引対象の受け側および払い側のそれぞれにおいて、該参照情報群から前記各単位取引モデル化機能を順次参照し前記時価評価演算を実行させてその演算結果を取得し、該各演算結果に基づいて前記金融取引の特性を算出する取引系列モデル化機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。

【請求項２６】　コンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

それぞれオプション取引の原資産を個別にモデル化し、それぞれ時価評価演算機能を有する１つ以上の原資産モデル化機能と、

該各原資産モデル化機能を参照するための参照情報群を保有し、該参照情報群から順次参照される前記各原資産モデル化機能内の前記時価評価演算による各演算結果および／または所定の評価モデルに基づいて前記オプション取引の特性を算出するオプションモデル化機能と、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、金融取引のリスク管理をコンピュータを用いて統合的に行うための金融リスク管理技術に関する。更に言えば、実際のあるいは架空の金融取引に関するパラメータを任意に設定し、組み合わせることにより、任意の金融商品を自由に設計可能であり、該金融取引の現時点あるいは将来における時価を算出することにより、架空の金融商品のシミュレーションあるいは実際の金融商品の運用（リスク管理）を高速に実行できる統合金融リスク管理装置、および該統合金融リスク管理装置に使用することができる金融取引モデル化装置に関する。

【０００２】

【従来の技術および発明が解決しようとする課題】

金融市場の多様化に伴って、金融取引も多様化・複雑化してきており、金融取引のリスク管理は金融関係者およびそのユーザにとって必須の要件となりつつある。

【０００３】

特に、解約権付きスワップやキャップ付きスワップなどの、複数の金融取引実体からなる複合取引に対する統合リスク管理技術が要請されている。

【０００４】

このような複合取引の基礎となる個々の取引実体のリスク管理のためには、従来は、個々の取引実体の種類に応じたリスク管理モジュールのプログラミングが必要であり、開発コストが膨大になると共に、新たな金融取引に対する柔軟かつ迅速な対応が困難であるという問題点を有していた。

【０００５】

典型的な従来のリスク管理システムとしては、あらゆる機能を実装したものがいくつか知られている。しかし、そのようなシステムは、例えば為替スポットや株式スポット、割引債、金利スワップ、クーポン・スワップ、通貨スワップ、あるいはオプションといった金融取引のすべての機能を実装する必要があるため、膨大な開発コストが必要となりその販売コストも膨大になってしまうという問題点を有していた。

【０００６】

現実の金融取引としては、「交換」をベースとするキャッシュ・フロー系の取引が大部分を占めているが、従来のシステムはこのような共通性を活用しておらず、全体として効率の良いシステムではなかったため、上述のようにコスト増を招いていた。

【０００７】

一方、複合取引に対する統合的なリスク管理技術として、紐付け管理と呼ばれる従来技術が知られている。

【０００８】

この従来技術においては、複合取引を構成する複数の取引実体に対して共通の

紐付け番号が付与され、データベース上でこの紐付け番号を使って複数の取引実体が１つの複合取引として一括してリスク管理される。

【０００９】

このような従来技術において、複合取引に対するリスク管理のための諸演算を実行するためには、紐付け番号を使って複合取引を構成する個々の取引実体をデータベース上で検索し、検索された個々の取引実体に対してリスク管理演算を実行してそれぞれの演算結果をデータベースに格納した後、何らかの方法によって実現される上記個々の取引実体間の関連付けに基づいてデータベース上で検索・集計を行う手続きが必要となる。

【００１０】

従って、従来は、複合取引に対する統合リスク管理システムを開発するためのプログラミングが膨大になり、開発コストの増大とシステムの肥大化を招くという問題点を有していた。

【００１１】

また、従来は、そのようにして開発された統合リスク管理システムが実行する演算処理量も膨大であり、統合リスク管理システムの性能の向上が困難であるという問題点を有していた。

【００１２】

さらに、複合取引を構成する個々の取引実体の種類および個数が増加するに従って、それらのデータを格納するための記憶容量も増大してしまうという問題点を有していた。

【００１３】

上記のような問題点を解決するために、従来は、特に金融取引を大量に行う金融機関等において、大容量かつ高性能なコンピュータの導入が必要となり、統合リスク管理のための設備投資および運用コストが膨大になってしまっていた。

【００１４】

本発明の課題は、シンプルなシステム構成を実現することにより、金融取引に対する統合リスク管理システムの開発・運用コストを低減させ、システム性能を向上させることにある。

【００１５】

【課題を解決するための手段】

まず、本明細書および図面における「クラス」「インスタンス」「メソッド」「コンテナ・クラス」等の用語は周知のオブジェクト指向概念における所定の機能、構成を意味する。なお、上記以外の用語もオブジェクト指向概念における所定の機能を意味する場合がある。また、「参照」とは、参照元のプログラムが、読み出すべきデータの格納アドレス情報を直接保持しており、該アドレス情報を利用して必要なデータを読み出すことを指す。

【００１６】

図１は、本発明の第１の態様の原理を示すブロック図である。

【００１７】

本発明の第１の態様は、金融に関する１つ以上の取引実体（スワップ取引、オプション取引等）からなる複合取引（１つの取引時点のみによって構成されてもよい）に対してリスク管理を行う統合金融リスク管理装置を前提とする。

【００１８】

１つ以上の取引実体モデル化手段１０３は、それぞれ取引実体に関する情報を個別に格納する格納手段１０２と、取引実体に関する時価評価演算手段１０１とを有する。この取引実体モデル化手段１０３は、例えば、格納手段１０２が取引実体をモデル化するためのパラメータをクラスメンバとして保持し、時価評価演算手段１０１をメソッド（GetNPVメソッド）として保持し、オブジェクト指向概念における所定のクラスのインスタンス（TContract インスタンス）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。

【００１９】

仮想取引手段１０７は、各取引実体モデル化手段１０３を参照するための参照情報群１０４を保有する参照情報記憶手段１０５と、所定の指示に基づいて、その参照情報群１０４から各取引実体モデル化手段１０３を順次参照し時価評価演算手段１０１を実行させてその演算結果を取得し、その各演算結果に基づいて複合取引の特性を算出する複合取引特性算出手段１０６とを有する。ここで、前述

の取引実体モデル化手段１０３は例えばリスト構造として実装され、仮想取引手段１０７は、参照情報記憶手段１０５が参照情報群１０４をリストメンバ（リンクリストLinkList）として保持し、複合取引特性算出手段１０６を仮想メソッド（GetNPV）として保持し、オブジェクト指向概念における所定のコンテナ・クラスのインスタンス（TVirtualContractインスタンス）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。

【００２０】

なお、金融特性算出手段１０８については、後述する。

【００２１】

上述の本発明の第１の態様の構成によれば、１つ以上の取引実体から構成される金融に関する複合取引（複合商品）に対するリスク管理において、従来同時に達成することが困難であった、システムの単純化・リスク管理演算の高速化・データ圧縮という３つの要請を同時に達成することが可能となる。

【００２２】

すなわち、本発明の第１の態様の構成では、対象となる取引の種類は問わずかつその構成要素が単一か複合かを問わずに、全ての取引を例外なく１つの仮想取引手段１０７によって取りまとめることができる。このため、アプリケーション・プログラマは、取引種や構成を意識することなく、プログラミングを行うことができる。

【００２３】

具体的には、取引実体モデル化手段１０３がTContract クラスのインスタンスなどとして実装され、仮想取引手段１０７がTVirtualContractクラスのインスタンスなどとして実装されることにより、全ての取引を常にTVirtualContractインスタンスなどとして認識することができる。

【００２４】

そして、アプリケーション・プログラマは、TContract クラスなどの構造は変更することなく、仮想取引手段１０７における複合取引特性算出手段１０６を実現するTVirtualContractインスタンスなどにおけるメソッドを個別に開発するだ

けで、様々なリスク管理機能を容易に開発することが可能となる。

【００２５】

また、本発明の第１の態様の構成では、１つ以上の取引実体モデル化手段１０３が、仮想取引手段１０７内のリンクリスト等の参照情報群１０４により参照される単純な構造を有し、かつ、仮想取引手段１０７に対して１つの指示を発行するだけで、参照情報群１０４を介して各取引実体モデル化手段１０３内の時価評価演算手段１０１が独立に時価評価演算を実行し、最後に仮想取引手段１０７内の複合取引特性算出手段１０６が各時価評価演算結果を取りまとめる構成を有する。このため、リスク管理における各種演算を、参照情報群１０４を介した各取引実体モデル化手段１０３に対するシーケンシャルなアクセス動作によって高速に実行することが可能となる。

【００２６】

具体的には、取引実体モデル化手段１０３を、時価評価演算手段１０１をGetNPVメソッドとして保持するTContract クラスのインスタンスなどとして実装し、仮想取引手段１０７を、参照情報群１０４をリンクリストとして保持すると共に、複合取引特性算出手段１０６を仮想メソッドGetNPVとして保持するTVirtualContractコンテナ・クラスのインスタンスなどとして実装することにより、オンメモリ上での高速リスク管理演算が容易に実現される。

【００２７】

さらに、本発明の第１の態様の構成では、金融取引は、仮想取引手段１０７が独立して存在する１つ以上の取引実体モデル化手段１０３を参照するという構成を有するため、例えばスワップ取引、オプション取引といった１つ１つの取引実体モデル化手段１０３を複数の仮想取引手段１０７に参照させることが可能となる。すなわち、１つの取引実体モデル化手段１０３のデータを、複数の金融取引のために使い回すことが可能となる。この結果、同様の金融取引を大量に取り扱うような金融機関などにおいて、大きなデータ圧縮効果を得ることが可能となり、コンピュータ資源の削減が可能となる。

【００２８】

この場合に、例えば、仮想取引手段１０７内の複合取引特性算出手段１０６に

、各演算結果に所定の変換係数を乗じて新たな演算結果を算出するような機能を持たせることにより、金融取引ごとに各取引実体モデル化手段１０３に対する金融特性の微妙な差異を吸収することが可能となる。

【００２９】

図２は、本発明の第２の態様の構成図である。

【００３０】

本発明の第２の態様は、所定の取引期間（所定の時点のみをも含む）２０１において金融取引対象（通貨以外の金融商品をも含む）の受け払いを行うことによって成立する取引（キャッシュ・フロー系取引）をモデル化する金融取引モデル化装置を前提とする。なお、本発明の第２の態様の構成は、上述の本発明の第１の態様の構成を前提とするものではないが、本発明の第１の態様の構成における取引実体モデル化手段１０３を実現する一構成である。

【００３１】

１つ以上の単位取引モデル化手段２０７は、金融取引対象の受け側（レシーブサイド）２０２および払い側（ペイサイド）２０３のそれぞれにおいて、所定の取引期間２０１を受けあるいは払い毎に分割して得られる１つ以上の単位取引期間（CashFlowLet ）２０４ごとに設けられ、それぞれ、単位取引に関する情報を個別に格納する単位取引情報格納手段２０６と、その単位取引期間２０４の時価評価演算手段２０５を有する。この単位取引モデル化手段２０７は、例えば、単位取引情報格納手段２０６が単位取引期間２０４の時価評価演算を行うためのパラメータをクラスメンバとして保持し、時価評価演算手段２０５（GetNPV）をメソッドとして保持し、オブジェクト指向概念における所定のクラスのインスタンス（TCFSwap インスタンス）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。上述の時価評価演算手段２０５は、例えば、その単位取引モデル化手段２０７に対応する単位取引期間２０４における、金利、収益率、または価格の３種類の指数のいずれかに基づいて計算される将来価値指数と、金利、収益率、および価格のいずれの指数にも依存しない評価値とに基づいて時価評価演算を実行する。また、上述の時価評価演算手段２０５は、例えば、その時価評価演算結果に対してさらに所定の

割引率を乗じて得られる現在価値をその時価評価演算結果とする割引率乗算手段を含む。

【００３２】

取引系列モデル化手段２１１は、単位取引モデル化手段２０７を参照するための参照情報群２０８を金融取引対象の受け側２０２および払い側２０３のそれぞれに対応して保有する参照情報記憶手段２０９と、所定の指示に基づいて、金融取引対象の受け側２０２および払い側２０３のそれぞれにおいて、その参照情報群２０８から各単位取引モデル化手段２０７を順次参照し時価評価演算手段２０５を実行させてその演算結果を取得し、その各演算結果に基づいて取引系列の特性を算出する取引系列特性算出手段２１０とを有する。ここで、前述の単位取引モデル化手段２０７は例えばリスト構造として実装され、取引系列モデル化手段２１１は例えば、金融取引対象の受け側２０２および払い側２０３のそれぞれにおける参照情報群２０８をリストメンバ（RecCashFlows, PayCashFlows）として保持し、取引系列特性算出手段２１０を仮想メソッド（GetNPV）として保持し、オブジェクト指向概念における所定のコンテナ・クラスのインスタンス（TCFSwap インスタンス）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。

【００３３】

なお、金融特性算出手段２１３については、後述する。

【００３４】

上述の本発明の第２の態様の構成によれば、為替、現金貸借、債権、株式、商品、金利・通貨スワップ、エクイティ・スワップ、コモディティ・スワップをはじめとする「交換」をベースとする多様な金融取引を、１つの取引系列モデル化手段２１１によって統一的にモデル化することが可能となる。より具体的には、キャッシュ・フロー系取引実体が、受け側２０２および払い側２０３のそれぞれにおける単位取引期間２０４ごとのキャッシュ・フロー要素（CashFlowLet ）の集合として管理され、各要素がそれらを参照する取引系列モデル化手段２１１によって取りまとめられる構成を有する。そして、この参照様式が多様化されることによって、各種の「交換ベースのキャッシュ・フロー系取引」を統一的にモデ

ル化することが可能となるのである。

【００３５】

この結果、金融取引モデル化装置のシステム構成を非常にコンパクトなものにすることが可能となり、開発・販売コストの削減に大きく貢献する。

【００３６】

ここで金融取引対象は、必ずしも通貨である必要はなく、為替、債権、株式、商品などであってもよい。この場合には、単位取引モデル化手段２０７における時価評価演算手段２０５は、それに対応する単位取引期間２０４において、金融取引対象の単位（例えば株数など）のもとでの、金利、収益率、または価格に準ずる時価比率を演算することにより、将来価値指数を算出することができる。すなわち本発明の第２の態様では、どのような単位を有する金融取引対象であっても、「交換ベースのキャッシュ・フロー系取引」を統一的に扱えるのである。

【００３７】

本発明の第２の態様の構成で、ユーザ等による日付情報の設定に基づいて、所定の取引期間２０１を受けあるいは払い毎に分割して１つ以上の単位取引期間２０４を算出し、金融取引対象の受け側２０２および払い側２０３のそれぞれにおいて、算出した単位取引期間２０４ごとに、単位取引モデル化手段２０７を生成し、受け側２０２および払い側２０３のそれぞれに対応する取引系列モデル化手段２１１内の参照情報群２０８を生成するユーザ・インタフェース手段２１２（TCFChain.BuildCFメソッド）をさらに含むように構成することができる。

【００３８】

この構成により、ユーザは、上述した「交換」をベースとする多様な金融取引を、統一したユーザ・インタフェースを介して設計することが可能となり、操作性の向上に大きく貢献する。

【００３９】

また、本発明の第２の態様の構成において、単位取引モデル化手段２０７ごとに、それがモデル化する金融取引のためのパラメータを変更し、その変更に応じて所定の指示を発行するユーザ・インタフェース手段２１２（マジック・シート）をさらに含むように構成することができる。

【００４０】

この構成により、ユーザは、交換ベースのキャッシュ・フロー系取引に対し、非常に詳細なカスタマイズを行うことが可能となる。

【００４１】

図３は、本発明の第３の態様の原理を示すブロック図である。

【００４２】

本発明の第３の態様は、金融に関するオプション取引をモデル化する金融取引モデル化装置を前提とする。なお、本発明の第３の態様の構成は、前述の本発明の第１の態様の構成を前提とするものではないが、本発明の第１の態様の構成における取引実体モデル化手段１０３を実現する一構成である。

【００４３】

１つ以上の原資産モデル化手段３０３は、それぞれオプション取引の原資産に関する情報を個別に格納する格納手段３０２と、その原資産に関する時価評価演算手段３０１を有する。この原資産モデル化手段３０３は例えば、格納手段３０２が原資産をモデル化するためのパラメータをクラスメンバとして保持し、時価評価演算手段３０１をメソッド（GetNPV）として保持し、オブジェクト指向概念における所定のクラスのインスタンス（TOption インスタンス）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。

【００４４】

オプションモデル化手段３０８は、各原資産モデル化手段３０３を参照するための参照情報群３０４を保有する参照情報記憶手段３０５と、その参照情報群３０４から順次参照される各原資産モデル化手段３０３内の時価評価演算手段３０１による各演算結果および／または所定の評価モデル（ブラック・ショールズ・モデルなど）３０６に基づいてオプション取引の特性を算出するオプション取引特性算出手段３０７とを有する。ここで、前述の原資産モデル化手段３０３は例えばリスト構造として実装され、オプションモデル化手段３０８は、参照情報群３０４をリストメンバ（Underlyings ）として保持し、オプション取引特性算出手段３０７を仮想メソッド（GetNPV等）として保持し、オブジェクト指向概念に

おける所定のコンテナ・クラスのインスタンス（TOption インスタンス）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。オプションモデル化手段３０８は、例えば、所定の指示の発行時に、オプション取引に対して設定されている日付がそのオプション取引のイン・ザ・マネーの状態である場合に、参照情報群３０４から各原資産モデル化手段３０３を順次参照し時価評価演算手段３０１を実行させてその演算結果を取得し、その各演算結果および所定の評価モデル３０６に基づいてオプション取引の特性を算出し、上記日付がオプション取引のアウト・オブ・ザ・マネーの状態である場合に、所定の評価モデル３０６のみに基づいてオプション取引の特性を算出する。

【００４５】

なお、金融特性算出手段２１３については、後述する。

【００４６】

従来のオプションのモデル定義では、オプション自体の評価モデルに原資産の特性を加味してオプションのモデルが決定されていた。これに対して本発明の第３の態様の構成では、原資産モデル化手段３０３は、オプションの評価モデルとからは完全に分離・独立して設計・実装することができ、オプションの本体であるオプションモデル化手段３０８は、参照情報群３０４を介して上述のようにして独立して実装される任意数の原資産モデル化手段３０３を参照することができる。すなわち、オプションモデル化手段３０８に対して１つの指示を発行するだけで、参照情報群３０４を介して各原資産モデル化手段３０３内の時価評価演算手段３０１が独立に時価評価演算を実行し、最後にオプションモデル化手段３０８内のオプション取引特性算出手段３０７が、各演算結果と所定の評価モデル（イン・ザ・マネー時）または所定の評価モデルのみ（アウト・オブ・ザ・マネー）に基づいて、オプション取引の特性を算出することができる。このため、本発明の第３の態様の構成によれば、オプション取引のシステム設計が簡略化され、性能の向上と開発・販売コストの削減に貢献する。

【００４７】

また、本発明の第３の態様の構成においては、１つの原資産モデル化手段３０

３のデータを、複数のオプション取引のために使い回すことが可能となる。この結果、同様のオプション取引を大量に取り扱うような金融機関などにおいて、大きなデータ圧縮効果を得ることが可能となり、コンピュータ資源の削減が可能となる。

【００４８】

さらに、本発明の第３の態様の構成においては、オプションモデル化手段３０８は、オプション取引に対して設定されている日付がそのオプション取引のイン・ザ・マネーの状態である場合に、各原資産モデル化手段３０３内の時価評価演算手段３０１による各演算結果と所定の評価モデル３０６に基づいてオプション取引の特性を算出し、上記日付がオプション取引のアウト・オブ・ザ・マネーの状態である場合に、所定の評価モデル３０６のみに基づいてオプション取引の特性を算出するように構成することができる。この結果、イン・ザ・マネー／アウト・オブ・ザ・マネーの別による切替え制御が単純化され、高性能なシステムを構築することが可能となる。

【００４９】

本発明の第４の態様は、上述した本発明の第１、第２、または第３の態様の構成を前提とする。

【００５０】

そして、図１、図２、または図３の金融特性算出手段１０８，２１３，または３０９は、各時価評価演算手段１０１，２０５，または３０１に対して、その各時価評価演算手段による各時価評価演算時の金融特性（リアルタイムデータを含む）を算出し、その金融特性を各時価評価演算手段１０１，２０５，または３０１に供給する。この金融特性算出手段１０８，２１３，または３０９は、例えば、上記金融特性をモデル化し、オブジェクト指向概念における所定のクラスのインスタンス（TCntYieldCurves,TCntPriceCurves,TCntVolsCurvesなど）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）である。そして、各時価評価演算手段１０１，２０５，または３０１は、上記所定のクラスのインスタンスが保有し、上記金融特性を算出するための所定のメソッドを参照する参照情報（DiscCurve,PriceCurve等）を含む

【００５１】

また、上述の本発明の第４の態様の構成において、上述の金融特性を定義する金融特性定義手段をさらに含むように構成することができる。

【００５２】

上述の本発明の第４の態様の構成によれば、ユーザは、金融取引対象の単位、休日除外都市、イールド・カーブ、プライス・カーブ、ボラリティ・カーブ等の各種金融特性を自由に時価評価演算に導入し、設計することも可能となる。

【００５３】

本発明の第４の態様の構成において、金融特性算出手段１０８，２１３，または３０９は、各時価評価演算手段１０１，２０５，３０１による各時価評価演算時の複数の単一金融特性をそれぞれ算出する複数の単一金融特性算出手段と、それら複数の単一金融特性を合成して新たな仮想金融特性を算出し、その仮想金融特性を各時価評価演算手段１０１，２０５，または３０１に供給する仮想金融特性算出手段とを含むように構成することができる。この場合、各単一金融特性算出手段は、例えば、各単一金融特性をモデル化し、オブジェクト指向概念における複数の所定のクラスのそれぞれのインスタンス（TAbsCurve ）を実行するモジュール（クラスモジュール５０１、ＲＤＢＭＳモジュール５０２、ＲＤＢモジュール５０３）であり、仮想金融特性算出手段は、例えば、複数の所定のクラスが継承する１つの所定のスーパークラスのインスタンス（TCntVirtualCurve）を実行するモジュールである。そして、各時価評価演算手段１０１，２０５，または３０１は、上記スーパークラスが保有し、仮想金融特性を算出するための所定のメソッドを参照するための参照情報を含む。

【００５４】

この構成によれば、ユーザは、複数の金融特性を合成してさらに効果的な金融特性を時価評価演算に導入することが可能となる。

【００５５】

なお、本発明は、コンピュータにより使用されたときに、上述の本発明の各態様の構成によって実現される機能と同様の機能をコンピュータに行わせるための

コンピュータ読出し可能記録媒体として構成することもできる。

【００５６】

【発明の実施の形態】

以下、図面を参照しながら本発明の実施の形態について詳細に説明する。

【００５７】

図４は、本発明の実施の形態の概念図、図５は、本発明の実施の形態の全体構成図である。以下、これらの図を随時参照しながら、本発明の実施の形態の概念および詳細について、順次説明する。

０．本発明の特徴

本発明の特徴は、以下の６点である。

【００５８】

１．複数の取引実体に対応するクラス（TCFSwap/TOption クラス）をとりまとめて１つの仮想取引を実現するTVirtualContractコンテナ・クラスの枠組み（仮想取引管理方式）

２．キャッシュ・フロー系取引実体を、受けサイド（レシーブサイド）および払いサイド（ペイサイド）のそれぞれにおける単位取引期間ごとのキャッシュ・フロー要素（CashFlowLet ）の集合として管理し、各CashFlowLet に対応するTCF クラスをとりまとめて１つのキャッシュ・フロー系取引実体を実現するTCFSwap コンテナ・クラスの枠組み

３．各TCF クラス（CashFlowLet ）での共通化された時価評価演算処理

４．各TOption （コンテナ）クラスの枠組み

５．金融カーブ定義機能

６．リスク管理のためのパラメータ変更とそれに対するシミュレーション結果の表示を容易に行うことのできるユーザ・インタフェース

以下、本発明に関する上記１．～６．の特徴のそれぞれについて、本発明の実施の形態に沿って詳細に説明する。

１．仮想取引を実現するTVirtualContractコンテナ・クラス

（仮想取引管理方式）

まず、本発明の第１の特徴であるTVirtualContractコンテナ・クラスについて説明する。

１．１　仮想取引の概念

本実施の形態では、図４に示されるように、金融取引は複数の取引実体から構成される複合取引（複合商品）であるのが一般的であるとの前提が仮定されている。

【００５９】

この前提に基づいて本実施の形態では、コンピュータ上において金融取引をオブジェクト指向概念に基づいてコード化し、オブジェクト指向概念に基づいて統合リスク管理を実現する。なお、本発明においては、オブジェクト指向概念は必須の要件ではないが、本発明に基づいてコンピュータ上で統合リスク管理システムを効率的に構築する助けとなる概念であるため、本実施の形態はオブジェクト指向概念に基づくシステム構築を例として説明する。

【００６０】

本実施の形態ではまず、図５に示されるように、各取引実体に１つずつ対応するTContract という抽象クラスが定義される。そして、これらの取引実体から構成される複合取引としての金融取引が、TContract インスタンスを任意数保持できるコンテナ・クラスTVirtualContractにより表現される。本実施の形態では、このようなオブジェクト指向概念によって制御される金融管理を、「仮想取引管理」と呼び、上述の複合取引を仮想取引と呼ぶ。

【００６１】

例えば、「解約権付きスワップ」のインスタンス・イメージにおいては、図６に示されるように、１つめの取引実体に対応するInterestRateSwapという名前のTContract インスタンスと、２つめの取引実体に対応するOptionOnSwapという名前のTContract インスタンスが、「解約権付きスワップ」という仮想取引に対応するTVirtualContractインスタンスの集合要素となる。

【００６２】

この仮想取引管理方式は、一見複雑であるが、人間が金融取引実体の集合を仮想的な複合取引として認識する手順をコンピュータ上に忠実に反映するものである。すなわち、この事例である解約権付きスワップのような複合取引を人間が認識する場合、まずそれがどのような取引であるのかを示す名前（解約権付きスワップ）すなわち集合名でその仮想取引を認識し、必要に応じてその構成要素（集合要素）である取引実体に関心を向けるはずである。

【００６３】

人間が無意識に行う、命名による集合の認識は、複雑な事象を単純化して扱うための手法である。そして、この手法は、単一取引と複合取引を区別することなく仮想取引として扱うという本実施の形態における機能要件を満足する上で、不可欠な基本概念となっている。すなわち、この概念では、金融取引の一般型が複合取引として把握される。従って、例えば、単純な定型取引の１つであるプレーンバニラ・スワップが表現される場合においても、図７に示されるように、集合要素がInterestRateSwapという名前の１件のみの取引実体である、InterestRateSwapという同一名の仮想取引として表現されることになる。

【００６４】

このように、本実施の形態では、単純取引も複合取引と同様に仮想取引コンテナに保持させることにより、多種多様な金融取引の統合リスク管理を単純なアルゴリズムで実現することが可能となる。

１．２　仮想取引を実現するクラス構造

本実施の形態における仮想取引の、オブジェクト指向概念上でのクラス構造を図５および図８に示す。

【００６５】

各図中、TCFSwap は、各種キャッシュ・フロー系取引のキャッシュ・フロー要素を保持するコンテナ・クラスである。また、TOption は、各種オプションを代表する抽象クラスを表している。このように、本実施の形態では、TContract クラスを継承する、各種取引実体に対応するクラスが定義される。

【００６６】

TVirtualContractクラスは、各種取引実体に対応するTCFSwap/TOption クラスのインスタンスを保持する機能を有するコンテナ・クラスで、TVirtualContractインスタンスは、そのメンバの１つとして、各TCFSwap/TOption インスタンスのリスト構造への参照であるリンクリストを保持している。

【００６７】

なお、TCFSwap およびTOption に関する各クラス構造については後述する。

【００６８】

図５および図８に示されるクラス構造が持つ優れた特性は、例えば図６に示されるような複合取引の時価評価の演算手順を想定すると、容易に理解することができる。

【００６９】

TContract クラスは、時価評価演算を実行する仮想メソッドGetNPV()を持ち、TCFSwap クラスとTOption クラスは共にこの仮想メソッドをオーバーライドしている。このため、TVirtualContractクラスのインスタンスであるCancelableSwap（解約権付きスワップ）インスタンスがアクティブになった段階で、そのインスタンスのリスト構造に格納されている、TContract クラスをそれぞれ継承する取引実体クラスTCFSwap およびTOption の各インスタンスInterestRateSwapおよびOptionOnSwapに、順次仮想メソッドGetNPV()が発行されることにより、解約権付きスワップの時価評価演算の解が得られる。

【００７０】

このように本実施の形態では、各TCFSwap インスタンスまたはTOption インスタンスがTVirtualContractインスタンス内のリンクリストを介して保持されることにより、金融取引の対象が単純取引であっても複合取引であって、従来の紐付け管理におけるような条件分岐を一切導入することなく統合リスク管理のための諸演算を実行できる。すなわち、本実施の形態は、オブジェクト指向概念における多態性が持つ高演算効率を最大限反映させたクラス構造を有する。

１．３　仮想取引の優位性

仮想取引管理方式の従来技術に対する優位性は、以下の３点である。

１．３．１　プログラム構造の単純化
１．３．２　複合取引に対する演算の高速化
１．３．３　データ圧縮

以下に、上記各優位性について、順次説明する。

１．３．１　プログラム構造の単純化

本実施の形態では、対象となる取引の種類は問わずかつその構成要素が単一か複合かを問わずに、全ての取引を例外なく１つのTVirtualContractインスタンスによって表現することができる。このため、アプリケーション・プログラマは、取引種や構成を意識することなく、プログラミングを行うことができる。

１．３．２　複合取引に対する演算の高速化

システムの構築手段としてオブジェクト指向概念を導入するかしないかに関わらず、仮想取引管理方式を導入しない場合には、複合取引に対する諸演算のために、個別取引実体に対する演算の後、何らかの方法によって実現される個別取引実体間の関連付けに基づいて検索・集計を行う手続きが必要となる。これに対して、仮想取引管理方式では、上記関連付けに相当する部分をリンクリストによって実現しているため、上記手続を、リンクリストに登録されている各ポインタをシーケンシャルにアクセスする手続きによって代替できる。

【００７１】

従って、対象となる複合取引の構成要素が多くなるほど、すなわちその複合取引が複雑になるほど、本実施の形態の演算性能と従来技術の演算性能との差は大きくなることが期待できる。

１．３．３　データ圧縮

仮想取引管理方式は、メモリあるいはデータベース上の取引属性データのサイズ

を圧縮する機能をも提供する。以下、単純化した事例によって、この機能を実現する手段について説明する。

【００７２】

金利スワップを大量に取り扱う金融機関において、日常的に発生しうるケースとして、図９に示されるような２つの金利スワップ・オブジェクトを生成する場合を想定する。なお、実際の金利スワップ契約を識別するためには、より詳細な属性情報が必要であるが、ここでは仮想取引のデータ圧縮効果を説明するのに必要な属性のみ設定されている。

【００７３】

これら２つの契約における相違点は、契約先と想定元本の属性内容である。なお、便宜上、１つ目の金利スワップをInterestRateSwap001 、２つ目の金利スワップをInterestRateSwap002 と命名しておく。

【００７４】

これら２つのオブジェクトの生成イメージを図１０に示す。この図は、２つのTVirtualContractインスタンスの各々がName,Counterparty,NotionalConvFactorという３つの属性データと、TCFSwap インスタンスInterestRateSwapへの参照を１つ保持しているリンクリスト（LinkList）とを有していることを示す。ここで、TVirtualContractインスタンスInterestRateSwap002 のリンクリストに保持されるInterestRateSwapインスタンスへの参照アドレス（SameAddress と表記されている部分）がデータ圧縮の鍵となる。ここには、TVirtualContractインスタンスInterestRateSwap002 のリンクリストに保持されるInterestRateSwapインスタンスへの参照アドレスと同一のアドレスが保持されることにより、TCFSwap インスタンスInterestRateSwapの共有化が実現され、データ圧縮が実現されるのである。

【００７５】

ただし、TVirtualContractインスタンスInterestRateSwap002 のリンクリストに保持される上述の参照アドレスをこのまま利用すると、InterestRateSwap002 インスタンスの想定元本が、InterestRateSwap001 インスタンスの想定元本と同じ１０億円になってしまう。このため本実施の形態では、各TVirtualContractク

ラスには、基準となる想定元本（ここではInterestRateSwap001 インスタンスの想定元本）を各インスタンスの想定元本に変換するための変換係数NotionalConvFactorが定義される。すなわち、InterestRateSwap001 インスタンスには変換係数NotionalConvFactorとして値１が保持され、InterestRateSwap002 インスタンスには変換係数NotionalConvFactorとして値２が保持される。

【００７６】

以上説明した事例は、契約期間、利払い日構造、および約定レートが同一で、契約先と想定元本が異なる場合においてのみデータ圧縮効果が得られる事例であるが、本実施の形態による仮想取引管理方式では、要求されるデータ圧縮効果に応じた機能拡張も可能である。例えば、先の事例において、２つの契約間で約定レートも異なる場合には、図１０に示されるTVirtualContractクラスに、約定レートの変換係数を保持する属性を定義すればよい。同様にして、変換係数あるいは関数によって再現可能な属性を増やすことにより、高いデータ圧縮効果を得ることが可能である。

【００７７】

本実施の形態の仮想取引管理方式によるデータ圧縮は、ハードウエア・リソースの節減にのみ貢献するものではなく、演算の高速化作用を合わせ持つ。

【００７８】

例えば、図９の事例に示される２件の金利スワップの金利感応度が計測される場合には、金利スワップを構成する各利払い日ごとに発生するキャッシュ・フロー要素の現在価値を更新する手続きが必要となる。この場合に、図１０に示される手法では、この更新手続きが、InterestRateSwap001 インスタンスに対してのみ実行され、InterestRateSwap002 インスタンスについては、上記更新手続きで得られた値に変換係数NotionalConvFactorを乗算することによってその更新手続きが完了する。一般的にデータ圧縮は、アプリケーションの動作速度に対してネガティブ・ファクタとなるが、本実施の形態の仮想取引管理方式ではポジティブ・ファクタとなることが特徴である。

１．４　ＲＤＢＭＳ／ＲＤＢによる仮想取引管理方式の実現手段

オブジェクト指向概念に基づいて実装されたプログラムにおいては、メモリ上に生成されたインスタンスの永続保持にはＯＯＤＢＭＳ（オブジェクトオリエンテッドデータベースマネージメントシステム）を利用するのが自然であるが、導入コスト、既存資産の活用、安定稼働実績等を考慮してＲＤＢＭＳ（リレーショナルデータベースマネージメントシステム）を採用する場合も多い。ここでは、インスタンスの永続保持にＲＤＢＭＳを用いる場合の仮想取引管理方式の実現手段を例示する。

【００７９】

仮想取引をＲＤＢＭＳで管理するには、ＲＤＢ上に、少なくとも以下の３つのインスタンスまたはリストを格納するための３つのテーブルが必要となる。

【００８０】

・TVirtualContractインスタンス

・TContract インスタンス

・TVirtualContractインスタンス内のリンクリスト

これらのテーブルの管理は、ＲＤＢＭＳが管理するＲＤＢ（リレーショナルデータベース）上で一般的に利用される第三正規化を適用することによって、図１１に示されるように容易に実現できる。すなわち、外部キーのみで構成される中間テーブルLinkListを導入することにより、TVirtualContractインスタンスのコンテナに格納される情報を記録することが可能になる。

【００８１】

図１２は、図５のＲＤＢＭＳモジュールによって管理され、図５のＲＤＢモジュールに記憶される、TVirtualContractインスタンスを格納するVIRTUALDEAL テーブル、TContract インスタンスを格納するINDIVDEAL テーブルとDEAL__KINDテーブル、およびTVirtualContractインスタンス内のリンクリストを格納するためのLINKLISTテーブルの各レコードのデータ構造図である。なお、図５に示されるＲＤＢモジュール上のTCFCHAINテーブル、TCF テーブル、および各OPTIONテーブルには、TContract クラスから継承されるTCFSwap クラスのインスタンス、その

インスタンスにリンクするTCF クラスのインスタンス、およびTContract クラスから継承されるTOption クラスのインスタンスが格納されるが、これらについては後述する。

【００８２】

VIRTUALDEAL テーブルは、TVirtualContractインスタンスを一意に識別するための各仮想取引インスタンス識別コード VIRTUAL__ID（図１１のVirtualID に対応する）が付与された各レコードに、TVirtualContractインスタンスの内容を保持する。図１２に示されるその他のフィールド名は、ユーザによる管理の便宜のために設けられている。

【００８３】

INDIVDEAL テーブルの各レコードは、TContract インスタンスを一意に識別するための実体取引インスタンス識別コード INDIV__ID（図１１のContractIDに対応）と、そのTContract インスタンスにつきユーザが自由に命名することのできる取引種名KIND（図１１のContractIDごとのNameに対応）を保持する。

【００８４】

DEAL__KINDテーブルの各レコードは、取引種名KINDと、それに対応するクラス・ライブラリ内のクラス種別（TCFSwap クラスまたはTOption クラス）を識別するためのクラス識別コードTYPEを保持する。

【００８５】

LINKLISTテーブルの各レコードは、仮想取引インスタンス識別コード VIRTUAL__IDと実体取引インスタンス識別コード INDIV__IDとを保持することによって、VIRTUALDEAL テーブルが保持するレコードと、INDIVDEAL テーブル・TCFCHAINテーブル／TCF テーブル・各OPTIONテーブルのそれぞれ対応するレコードとを関連付ける。この場合、TContract クラスから継承される各TCFSwap クラスのインスタンスとそのインスタンスにリンクする各TCF インスタンス、およびTContract クラスから継承される各TOption クラスのインスタンスの各実体情報は、上述の INDIV__IDを介して、後述するTCFSwap テーブルとTCF テーブル、および各OPTIONテーブルに保持される。

【００８６】

上述のテーブル構成により、LINKLISTテーブルを介して、INDIVDEAL テーブル上の実体取引インスタンス識別コード INDIV__IDで識別される１つのTContract インスタンス（TCFSwap/TOption インスタンス）を、VIRTUALDEAL テーブル上の複数のTVirtualContractインスタンスに関連付けることが可能となり、図９および図１０で説明したデータ圧縮が可能となる。

【００８７】

なお、前述したように、データベース管理システムとしてＯＯＤＢＭＳを採用する場合には、図５および図１２に示されるようなテーブル構成を採用する必要はなく、図５および図８に示されるクラス構造を直接ＯＯＤＢに格納することが可能となる。

【００８８】

図５および図１２に示されるＲＤＢモジュールに登録されている１つの仮想取引に対する時価評価演算は、以下のようにして実行される。

ステップ１：図５のＲＤＢＭＳモジュールが管理するデータアクセスメソッドの１つであるBookOut メソッドが起動されることにより、VIRTUALDEAL テーブルから指定された識別IDを含むレコードが抽出され、そのレコードから仮想取引インスタンス識別コード VIRTUALDEAL.VIRTUAL__IDが取得される。

ステップ２：LINKLISTテーブルから、上述の仮想取引インスタンス識別コードを含むレコードが抽出されて、そのレコードから実体取引インスタンス識別コード LINKLIST.INDIV__IDが取得される。１つの仮想取引インスタンス識別コードを含むレコードは、複数レコードが存在し得る。この場合には、複数の LINKLIST.INDIV__IDコードが取得される。次に、INDIVDEAL テーブルから、各実体取引インスタンス識別コードを各レコードが抽出され、その各レコードから各取引種名INDIVDEAL.KINDが取得される。

ステップ３：DEAL__KINDテーブルから、上記各取引種名に対応する各レコードが抽出され、その各レコードから各クラス識別コードDEAL__KIND.TYPE が取得され

る。

ステップ４：ステップ２でLINKLISTテーブルから取得された各実体取引インスタンス識別コード LINKLIST.INDIV_IDを含む各レコードが、TCFCHAINテーブルとTCF テーブル、または各OPTIONテーブルから抽出され、各レコードに含まれる各実体情報と、ステップ３で取得された各クラス識別コードに基づきクラス・ライブラリから取出された各クラス定義情報とから、メモリ上に、各TCFSwap インスタンスと各TCF インスタンス、または各TOption インスタンスが生成される。

ステップ５：ステップ１で抽出されたVIRTUALDEAL テーブル上のレコードの内容に基づいて、メモリ上に、TVirtualContractインスタンスが生成され、そのインスタンス内のリンクリストに、上記ステップ４で生成された各TCFSwap インスタンスまたは各TOption インスタンスへの参照アドレスが登録される。

ステップ６：上述のリンクリストから参照される各TCFSwap インスタンスまたはTOption インスタンスの仮想メソッドGetNPV()が実行される。

【００８９】

このようにして、本実施の形態では、複合取引がどのような取引実体によって構成されていようとも、１つの仮想メソッドGetNPV()を実行するだけで、その複合取引に対する時価評価演算が実現される。

【００９０】

なお、ＯＯＤＢＭＳが採用される場合は、上記ステップ１～５の操作は必要なく、上記ステップ６を直接実行することができる。

２．キャッシュ・フロー系取引実体を実現するTCFSwap コンテナ・クラス

次に、本発明の第２の特徴であるTCFSwap コンテナ・クラスの枠組みについて説明する。

２．１　TCFSwap コンテナ・クラスの特徴

図８で説明したように、本実施の形態では、TContract クラスを継承するキャッシュ・フロー系取引実体に対応するクラスとして、TCFSwap コンテナ・クラスが定義される。

【００９１】

TCFSwap コンテナ・クラスの最大の特徴は、為替、現金貸借、債権、株式、商品、金利・通貨スワップ、エクイティ・スワップ、コモディティ・スワップをはじめとする「交換」をベースとする多様な金融取引が、このクラス１つで実現されることである。従来技術では、これらの取引に対して、それぞれ対応するクラスが定義されるのが一般的であり、実装されている取引実体クラスに該当しない取引が新たに取り扱い対象とされる場合には、そのためのクラスを追加実装する（新たに開発する）必要があった。これに対して本実施の形態では、キャッシュ・フロー系取引実体が、受けサイド（以下、レシーブサイドと呼ぶ）および払いサイド（以下、ペイサイドと呼ぶ）のそれぞれにおける単位取引期間ごとのキャッシュ・フロー要素（以下、CashFlowLet と呼ぶ）の集合として管理され、各CashFlowLet に対応するTCF クラスをとりまとめるTCFSwap コンテナ・クラスが用意される。そして、TCFSwap コンテナ・クラスへの各TCF クラスの格納様式が多様化されることによって、各種の「交換ベースのキャッシュ・フロー系取引」を実現することが可能となる。

２．２　TCFSwap コンテナ・クラスの構造

TCFSwap は、CashFlowLet の集合を格納するコンテナ・クラスで、そのインスタンスのイメージは図１３に示される。

【００９２】

図中、RecCashFlowsとPayCashFlowsという名前の２つのTCFChainクラスのインスタンスは、それぞれまず、CashFlowLet であるTCF クラスのインスタンスの集合を格納するリンク構造への参照であるリンクリストを保持する。RecCashFlows インスタンスは、受け側（以下、レシーブサイドと呼ぶ）のCashFlowLet 群であるTCF インスタンス群に対するリンクリストを保持し、PayCashFlowsインスタン

スは、払い側（以下、ペイサイドと呼ぶ）のCashFlowLet 群であるTCF インスタンス群に対するリンクリストを保持する。

【００９３】

また上記２つのTCFChainクラスのインスタンスは、それぞれ、PricingEngineという名前のTPwrGadgetクラスからそれぞれ継承される、DiscCurve という名前のTCntYieldCurveクラスのインスタンスと、PriceCurveという名前のTCntPriceCurveクラスのインスタンスへの参照を保持する。TPwrGadgetクラスは、CashFlowLet ごとの時価評価演算を実現するための金融特性の演算機能を提供するクラスであり、イールド・カーブ（ディスカウント・カーブ）、プライス・カーブ（価格カーブ）、ボラティリティ・カーブ、通貨・商品・証券等の取引単位、および休日除外対象都市に関する情報を任意に定義・生成・保持することができる。図１３には、これらのうちイールド・カーブとプライス・カーブのみを示してある。すなわち、TCntYieldCurveクラスは、イールド・カーブ（ディスカウント・カーブ）を定義するためのクラスであり、そのクラスに実装されているGetFactorメソッドにより、要求日付におけるディスカウント・ファクターを取得することができる。TCntPriceCurveクラスは、プライス・カーブを定義するためのクラスであり、そのクラスに実装されているGetFactor メソッドにより、要求日付における価格指数を取得することができる。なお、TPwrGadgetクラスは、仮想カーブ機能も有するが、これについては後述する。

【００９４】

図１４は、TCFChainクラスにクラスメンバ群として保持される属性群の詳細を示す図である。図１４において、CFList属性は図１３のTCF インスタンス（CashFlowLet ）のリンク構造へのリンクリストに対応し、DiscCurve 属性は図１３のDiscCurve （TCntYieldCurve）インスタンスへの参照に対応し、PriceCurve属性は図１３のPriceCurve（TCntPriceCurve）インスタンスへの参照に対応する。図１４に示されるその他の属性群は、レシーブサイドまたはペイサイドのそれぞれにおけるキャッシュ・フローの全体的な特性を決定するためのものである。

【００９５】

これらの属性群は、後述するユーザ・インタフェースを介して設定される。

【００９６】

これらの属性群は、主として、金利・通貨スワップの取引属性に類似していることが解るはずである。これは、為替、現金貸借、債券、株式、商品等のスポット、フォワード取引もスワップの一形態と考えていることに他ならず、実際殆どのキャッシュ・フローの交換に基づく取引は、TCF インスタンスの集合様式と、このインスタンスの状態を決定する属性で表現できるのである。

【００９７】

図１５は、図５のＲＤＢモジュールに格納される、TCFChainクラスに対応するTCFCHAINテーブルの各レコードのデータ構造を示す図である。図１４に示されるクラスメンバ名と図１５に示されるフィールド名とで、大文字と小文字の区別を除いて同じスペルを有するものが同じ情報を保持する。ただし、図１５で、UNIT#CODE フィールドは図１４のAppliedUnit メンバに対応する。また、図１５で、INDIV__IDフィールドには、実体取引インスタンス識別コードが格納され、これにより図１２に示されるLINKLISTテーブルまたはINDIVDEAL テーブルとTCFCHAINテーブルとがリンクされ、前述の１．４項での時価評価演算のステップ４の処理が実現される。また、図１５で、TCFCHAIN__IDフィールドには、TCFChainインスタンスの識別コードが格納される。また、図１５で、SIDE__IDフィールドには、レシーブサイド／ペイサイドの区分コードが格納される。さらに、CashFlowLetリンクリストを示す図１４に示されるCFList属性は、メモリ上へのTCFChainインスタンスの展開時に、INDIV__IDフィールドおよびSIDE__IDフィールドの値に基づいて動的に生成されるものであるため、その属性に対応するフィールドは図１５のTCFCHAINテーブルには存在しない。

【００９８】

次に図１６は、図１３中の右側のTCF クラスの枠内に示されるクラスメンバ群として保持される属性群の詳細を示す図であって、単位取引期間ごとのキャッシュ・フロー要素についての金利、収益率、価格の各指数計算を実行するために必要十分なものが定義されている。また、図１７は、図５のＲＤＢモジュールに格納されるTCF クラスに対応するTCF テーブルの各レコードのデータ構造を示す図である。図１６に示される属性名と図１７に示されるフィールド名とで、大文字

と小文字の区別を除いて同じスペルを有するものが同じ情報を保持する。また、図１７で、 TCF__IDフィールドには、TCF インスタンスの識別コードが格納される。さらに、図１７で、 INDIV__IDフィールドには実体取引インスタンス識別コードが格納されるとともに、TCFCHAIN__IDフィールドにはそのTCF インスタンスがリンクするTCFChainインスタンスのサイド（レシーブサイド／ペイサイド）の区分コードが格納される。これにより図１２に示されるLINKLISTテーブルまたはINDIVDEAL テーブルとTCFCHAINテーブルとがリンクされ、前述した１．４項での時価評価演算のステップ４の処理が実現される。また、図１６に示されるTotalCF#PV属性とDF属性は、メモリ上へのTCF インスタンスの展開時に動的に生成されるものであるため、それらの属性に対応するフィールドは図１７のTCF テーブルには存在しない。

【００９９】

ここで、図１６において、NotionalAmnt、NonIndexCF#FV 、TotalCF#FV、およびTotalCF#PVは、必ずしも金額ではなく、場合によっては株式、商品の取引分量を保持することもできる。

【０１００】

なお、TCF クラスの仕様としては、必ずしも図１３および図１６に示されるクラス構造に限定されるものではない。例えば、TCF クラスを基底クラスとし、これを継承して金利、収益率、価格の各指数計算に対応するクラスを用意する方法も有り得る。つまり、そのインスタンス集合で多様な金融取引を表現できるのであればよい。

２．３　TCFSwap/TCFChain/TCFインスタンスの生成

TCFSwap/TCFChain/TCFインスタンスの生成手順は、定型取引の生成手順と非定型取引の生成手順の２種類に分類される。

【０１０１】

定型取引の生成手順を、図１８の説明図および図１９の処理フローに基づいて説明する。

【０１０２】

ユーザが、後述するユーザ・インタフェースを介して、キャッシュ・フロー系取引の生成を指示することによって、図１８に示されるTCFSwap インスタンスがまず生成され、続いて、そのTCFSwap インスタンスのメソッドによって、図１８に示される、RecCashFlowsおよびPayCashFlowsの２つのTCFChainインスタンスが生成され、さらに、各TCFChainインスタンスのコンストラクタによって、各TCFChainインスタンス内に、TCF インスタンス（CashFlowLet ）のリンク構造へのリンクリストとして用いられる図１８に示されるCFListという名前のTList インスタンスが生成される。

【０１０３】

次に、ユーザは、後述するユーザ・インタフェースを介して、TCFSwap インスタンスに保持される2つのTCFChainインスタンスに、図１４に示される各属性値のうち必要なものを与える。

【０１０４】

その後、ユーザは、後述するユーザ・インタフェースを介して、BuildCF メソッドの実行命令を発行する。この結果、図１８に示されるように、RecCashFlowsおよびPayCashFlowsの２つのTCFChainインスタンスごとに、必要に応じて１つ以上のTCF インスタンスが生成され、上記２つのTCFChainインスタンスの何れかに属するリンクリストCFListにリンク(Add(CF))される。

【０１０５】

図１９は、TCFSwap インスタンスを介してRecCashFlowsおよびPayCashFlowsの各TCFChainインスタンスに対して発行されるBuildCF メソッドの処理フローを示す図である。

【０１０６】

まず、ステップ１のGenerate Dates処理では、受払日、計算期間開始日／終了日等、取引に必要な日付群を生成する手続きが実行される。金融取引に必要な日付群の生成アルゴリズムは、周知のものであるためそのアルゴリズムの詳細については省略する。その概要としては、ユーザによって設定されたTCFChainインスタンスのPaymCenters,FixCenters,EDate,TDate,FDate,BDConv,EndEnd,AdjTDate,CFFrq,ResetFrq,FixingOffset,FixingBasis,およびPayIn の各属性値（図１４）

に基づいて、各キャッシュ・フロー要素の期間情報が算出される。

【０１０７】

次に、ステップ２では、図１４に示されるCF Freq 属性値として０より大きい値が設定されているか否かが判定される。ユーザは、後述するユーザ・インタフェースにおいて、各種スワップなどのキャッシュ・フロー系の取引を作成するときには受払頻度として０より大きい値を設定し、その他の為替等の単純売買の取引を作成するときには受払頻度として０を設定する。

【０１０８】

ステップ２でCF Freq 属性値が０より大きいと判定された場合には、ステップ３～６によって初回と最終回を除く各CashFlowLet に対応する各TCF インスタンスが生成される。

【０１０９】

まず、ステップ３では、ステップ１のGenerate Dates処理によって生成された期間情報と、ユーザによって設定された受払頻度情報（CF Freq 属性）とに基づいて受払回数（初回と最終回を除くCashFlowLet の要素数）が算出され、それがTCFChainクラス内のメンバ変数CFCount に設定される。

【０１１０】

次に、ステップ５で上記メンバ変数CFCount の値が１ずつ減算されながら、ステップ６でその値が０になったと判定されるまで、ステップ４で、TCFChainインスタンスに設定されている属性値と算出される各指数値とに基づいて、１つ以上のTCF インスタンスが生成される。

【０１１１】

具体的にはまず、ユーザにより設定された現サイド（レシーブサイドまたはペイサイド）のTCFChainインスタンスのFloatIndex,NotionalAmnt の各属性値（図１４）が、新たに生成されたTCF インスタンスのFloatIndex,NotionalAmnt の各属性値（図１６）として設定される。

【０１１２】

次に、ステップ１のGenerate Dates処理によって生成された各キャッシュ・フロー要素の期間情報と、ステップ５で順次更新されるTCFChainインスタンスのメ

ンバ変数CFCount の値に基づき、新たに生成されたTCF インスタンスに対して、DateFrom,DateTo,PreFixing,Fixing,Days,Years,およびPaymDateの各属性値（図１６）が算出されて設定される。

【０１１３】

次に、ユーザにより設定された現サイド（レシーブサイドまたはペイサイド）のTCFChainインスタンスのIndexVal属性値（図１４）に基づいて、新たに生成されたTCF インスタンスのIntRate またはMarginの各属性値（図１６）が設定される。また、Weight属性のデフォルト値は１であり、この値は後述するマジック・シートを用いて個別に変更できる。

【０１１４】

新たに生成されたTCF インスタンスの属性値NonIndexCF__FV（図１６）には０が設定される。

【０１１５】

また、新たに生成されたTCF インスタンスに設定された属性値PaymDateを引数として、ユーザにより設定されたTCFChainインスタンスの属性値DiscCurve （図１４）に対応するDiscCurve インスタンスのGetFactor メソッドが実行されることにより、新たに生成されたTCF インスタンスの属性値DFが算出されて設定される。

【０１１６】

新たに生成されたTCF インスタンスのPrePrice,Price,Return, IndexCF__FV, MarginCF__FV,TotalCF__FV,TotalCF__PVの各属性値は、後述するTCF インスタンスの更新手順によって計算される。

【０１１７】

新たに生成されたTCF インスタンスのUpdateCFメソッドは、後述するTCF インスタンスの更新手順を実行するためのメソッドである。

【０１１８】

ステップ４において実行される上述のTCF インスタンスの生成動作が、ステップ４～６のループ処理によって繰り返し実行されることによって、現サイド（レシーブサイドまたはペイサイド）のTCFChainインスタンスにリンクすべき全ての

TCF インスタンスが生成される。

【０１１９】

上記ループ処理の結果、ステップ６でメンバ変数CFCount の値が０になったと判定された場合、または前述のステップ２でユーザによって設定された受払頻度に対応する属性値CF Freq が０であると判定された場合には、ステップ７で、ユーザによってTCFChainインスタンスの属性値FinPrincAmntが設定されているか否か（Nullであるか否か）が判定される。

【０１２０】

ステップ７の判定が真（True）である場合には、ステップ８で最終回のTCF インスタンスが生成される。この生成方法は、ステップ４の場合と概ね同じであるが、ユーザにより設定されたTCFChainインスタンスの属性値FinPrincAmnt（図１４）が、新たに生成されたTCF インスタンスの属性値NonIndexCF__FV（図１６）として設定される処理が追加される。

【０１２１】

ステップ７の判定が偽（False)である場合には、ステップ９で、ユーザによってTCFChainインスタンスの属性値IniPrincAmntが設定されているか否か（Nullであるか否か）が判定される。

【０１２２】

ステップ９の判定が偽（False)である場合は、特には図示しないが、エラーとなる。

【０１２３】

ステップ９の判定が真（True）である場合には、ステップ１０で初回のTCF インスタンスが生成される。この生成方法は、ステップ４の場合と概ね同じであるが、ユーザにより設定されたTCFChainインスタンスの属性値IniPrincAmnt（図１４）が、新たに生成されたTCF インスタンスの属性値NonIndexCF__FV（図１６）として設定される処理が追加される。

【０１２４】

ステップ８またはステップ１０の処理の後、ステップ１１で、現サイド（レシーブサイドまたはペイサイド）のTCFChainインスタンスでUpdateCFs メソッドが

実行される。このメソッドが実行されると、そのTCFChainインスタンスにリンクしている全てのTCF インスタンスに対して、時価評価演算を実行させるための後述するUpdateCFメソッドが発行される。

【０１２５】

多くの銀行間取引は定型化されているため、キャッシュ・フロー系取引に関してはその種別によらずに、ユーザは後述する共通のユーザ・インタフェースを介して必要なパラメータを設定するだけで、上述した一連の処理によって、その取引オブジェクトを簡単に生成することが可能となる。

【０１２６】

一方、ユーザは、非定型取引を生成したい場合には、まずTCFChainインスタンスを上述の定型取引の場合と同様の手順で生成した後に、後述するマジック・シートと呼ばれるユーザ・インタフェースを使って、目的とする取引の性質を反映させるようにTCFインスタンス群の属性を簡単に変更し、またはTCFインスタンスを個別生成してTCFChainインスタンスに追加することができる。この場合、変更または追加後には、ユーザは明示的に前述したUpdateCFs メソッドの実行を指示することになる。

【０１２７】

このような非定型取引の代表例は、想定元本の増減額方式に規則性のないアモチ／アキュムレーション・スワップである。ただし、この場合も利払日、計算期間等の構成には規則性が存在する場合が殆どであるため、ユーザは、BuildCFメソッドを発行後、規則に該当しないTCFインスタンスの属性のみを修正することで足りるはずである。従って、TCFインスタンスを個別生成するケースは、極めて不規則なキャッシュ・フロー構成の取引に限定される。

【０１２８】

以上示したように、TCFChainクラスは、実際の金融取引で利用頻度の高い定型取引のためのキャッシュ・フロー集合を自動生成するためのテンプレートとしても機能することがわかる。また、TCFChainクラスの役割は、あくまでTCFインスタンス集合を一括生成することであり、ここに保持される属性値がリスク計測の諸演算に利用されることはない。そして、TCFChainクラスはキャッシュ・フロー

の交換に基づく多様な取引の差異を吸収するためのクラスであるということができる。

３．TCF インスタンスの更新メソッド：

TCF クラスに実装されるUpdateメソッドは、そのクラスが対応する単位取引期間における時価評価演算を実行する機能である。このメソッドは、例えばユーザが後述するユーザ・インタフェースを介して任意のパラメータを変更し、リスク管理計算を実行し直す必要が生じたときに、TVirtualContractインスタンスまたはTCFSwap インスタンスにおいてGetNPV()メソッドの実行が指示されるタイミングにおいて、TCFChainクラスのUpdateCFs メソッドを介して起動される。すなわち、本実施の形態では、単位取引期間ごとの時価評価演算は、各単位取引期間に対応するTCF クラスのUpdateメソッドが独立して実行することになる。

【０１２９】

このUpdateメソッドは、単位取引期間における、金利、収益率、または価格の３種類の指数のいずれかに基づいて計算される将来価値指数と、金利、収益率、および価格のいずれの指数にも依存しない評価値とに基づいて時価評価演算を実行し、その演算結果に対してさらに所定の割引率を乗じて得られる現在価値をその時価評価演算結果とする。

【０１３０】

このメソッドの特徴は、従来取引種ごとに規定されていたキャッシュ・フローの更新アルゴリズムが、３種類の指数「金利／収益率／価格」だけで表現されることにある。いずれの指数が指定された場合も、最終的にはDateFrom（計算期間開始日）とDateTo（計算期間終了日）の時価比率でキャッシュ・フローを決定できる。

【０１３１】

また、ここで金融取引対象は、必ずしも通貨である必要はなく、為替、債権、株式、商品などであってもよい。この場合には、上記Updateメソッドは、それに対応する単位取引期間において、金融取引対象の単位（例えば株数など）のもとでの、金利、収益率、または価格に準ずる時価比率を演算することにより、将来

価値指数を算出することができる。すなわち本実施の形態では、どのような単位を有する金融取引対象であっても、「交換ベースのキャッシュ・フロー系取引」を統一的に扱えるのである。

【０１３２】

図２０は、１つのTCF クラス（以下、現TCF クラスと呼ぶ）のUpdateメソッドの処理フローである。

【０１３３】

まずステップ１では、後述するユーザ・インタフェースを介してユーザによって指定された現TCF クラスが属するサイドのTCFChainクラスのIndexType 属性値（図１４参照）が、金利（IntRate ）、収益率（Return）、価格（Price ）のいずれであるかが判定される。

【０１３４】

IndexType 属性値が金利（IntRate ）であると判定された場合には、まずステップ２で、現TCF クラスの前期価格指数属性値PrePrice（図１６参照）として１がセットされる。

【０１３５】

次に、ステップ３では、後述するユーザ・インタフェースを介してユーザにより、現TCF クラスが属するサイドのTCFChainクラスのFloatIndex属性値（図１４参照）が設定されているか否かが判定される。

【０１３６】

FloatIndex属性値が設定されていない場合は、ステップ４で、現TCF クラスにおいて、IntRate 属性値にYears 属性値が乗算され、その乗算結果が現TCF クラスの当期価格指数属性値Price とされる。ここで、現TCF クラスのIntRate 属性値は、現TCF クラスが属するサイドのTCFChainクラスのIndexVal属性値（図１４参照）からコピーされたものである。このIndexVal属性値は、後述するユーザ・インタフェースを介してユーザにより設定される。また、Years 属性値は、前述の図１９のステップ４において現TCF クラスに対して設定された、現TCF クラスに対応する単位取引期間（計算期間）の年数表示を示すものである。このYears 属性値は、後述するユーザ・インタフェースを介してユーザにより設定された、

現TCF クラスが属するサイドのTCFChainクラスのPaymCenters,FixCenters,EDate,TDate,FDate,BDConv,EndEnd,AdjTDate,CFFrq,ResetFrq,FixingOffset,FixingBasis,およびPayIn の各属性値（図１４）に基づいて、前述した図１９のステップ１のGenerate Dates処理において計算される。

【０１３７】

このようにしてユーザが固定金利を指定した場合は、ユーザが指定した固定金利値IntRate に計算期間年数Years を乗じて得られる値が現TCF クラスの当期価格指数Price とされる。そしてステップ１０で、現TCF クラスの当期価格指数属性値Price を現TCF クラスの前期価格指数属性値PrePriceで除算して得られる値が、現TCF クラスにおける時価比率属性値Return（図１６参照）として得られる。この場合、現TCF クラスの前期価格指数PrePriceにはステップ２で１がセットされているため、結局単純に、ユーザが指定した固定金利値IntRate に現TCF クラスの計算期間年数Years を乗じて得られる現TCF クラスの当期価格指数Priceが、そのまま現TCF クラスの時価比率属性値Returnとされることになる。

【０１３８】

一方、ステップ３で、ユーザによりFloatIndex属性値が設定されていると判定された場合には、ステップ５で、現TCF クラスの計算期間終了日DateToにおけるプライス・カーブの値を、現TCF クラスの計算期間開始日DateFromにおけるプライス・カーブの値で除算して得られる値が、現TCF クラスの当期価格指数属性値Price とされる。各プライス・カーブの値は、計算期間終了日DateToまたは計算期間終了日DateToを引数として、現TCF クラスが属するサイドのTCFChainクラスが参照しているPriceCurveインスタンス（図１３参照）のGetFactor メソッドを呼び出すことにより、算出することができる。プライス・カーブは、ユーザが後述するユーザ・インタフェースを介して定義・指定することができる。

【０１３９】

このようにして、ユーザが変動金利を指定した場合は、現TCF クラスに対応する単位取引期間（計算期間）におけるユーザが指定したプライス・カーブの時価比率が、現TCF クラスの当期価格指数Price とされる。そしてステップ１０で、現TCF クラスの当期価格指数属性値Price を現TCF クラスの前期価格指数属性値

PrePriceで除算して得られる値が、現TCF クラスにおける時価比率属性値Returnとして得られる。この場合、前期価格指数PrePriceにはステップ２で１がセットされているため、結局、プライス・カーブから上記単位取引期間における時価比率として得られる現TCF クラスの当期価格指数Price が、そのまま現TCF クラスの時価比率属性値Returnとされることになる。

【０１４０】

前述したステップ１で、ユーザによって指定されたIndexType 属性値が収益率（Return）であると判定された場合には、まずステップ８で、現TCF クラスの計算期間開始日DateFromにおけるプライス・カーブの値が、現TCF クラスの前期価格指数属性値PrePriceの値とされる。プライス・カーブ値の計算方法は、前述したステップ５の場合と同じである。

【０１４１】

次に、ステップ９で、現TCF クラスの計算期間開始日DateToにおけるプライス・カーブの値が、現TCF クラスの当期価格指数属性値Price の値とされる。プライス・カーブ値の計算方法は、前述したステップ５の場合と同じである。

【０１４２】

その後、ステップ１０で、現TCF クラスの当期価格指数属性値Price を現TCF クラスの前期価格指数属性値PrePriceで除算して得られる値が、現TCF クラスにおける時価比率属性値Returnとして得られる。

【０１４３】

前述したステップ１において、ユーザによって指定されたIndexType 属性値が価格（Price ）であると判定された場合は、まずステップ６で、現TCF クラスの前期価格指数属性値PrePriceとして１がセットされる。

【０１４４】

次に、ステップ７では、現TCF クラスの計算期間開始日DateToにおけるプライス・カーブの値が、現TCF クラスの当期価格指数属性値Price の値とされる。プライス・カーブ値の計算方法は、前述したステップ５の場合と同じである。

【０１４５】

その後、ステップ１０で、現TCF クラスの当期価格指数属性値Price を現TCF

クラスの前期価格指数属性値PrePriceで除算して得られる値が、現TCF クラスにおける時価比率属性値Returnとして得られる。この場合、前期価格指数PrePriceにはステップ６で１がセットされているため、結局、現TCF クラスの計算期間終了日DateToにおけるプライス・カーブの値が、そのまま現TCF クラスの時価比率属性値Returnとされることになる。

【０１４６】

以上のようにして、ユーザが金利、収益率、価格のどの指数を指定しても、最終的には、現TCF クラスの単位取引期間における時価比率Returnが得られることになる。

【０１４７】

次に、図２０のステップ１１においては、上述のようにして計算された時価比率Returnに現TCF クラスにおける想定元本金額属性値NotionalAmntを乗じて得られる値が、現TCF クラスにおける指数に基づく受払金額属性値 IndexCF__FV（図１６参照）とされる。現TCF クラスにおける想定元本金額属性値NotionalAmntは、後述するユーザ・インタフェースを介してユーザによって設定された、現TCF クラスが属するサイドのTCFChainクラスの想定元本額属性値NotionalAmntからコピーされたものである。

【０１４８】

続いて、ステップ１２では、現TCF クラスにおけるマージン値属性値Marginに現TCF クラスにおける想定元本金額属性値NotionalAmntを乗じて得られる値が、現TCF クラスにおけるマージンに基づく受払金額属性値MarginCF__FV（図１６参照）とされる。現TCF クラスのマージン値属性値Marginは、現TCF クラスが属するサイド（レシーブサイドまたはペイサイド）のTCFChainクラスのIndexVal属性値からコピーされたものである。ここでユーザは、後述するユーザ・インタフェースを介して現TCF クラスが属するサイドに対し変動金利を指定した場合に、当該サイドのTCFChainクラスのIndexVal属性値としてマージン値を指定できる。

【０１４９】

さらに、ステップ１３では、ステップ１１で算出された現TCF クラスにおける指数に基づく受払金額属性値 IndexCF__FVに現TCF クラスのWeight属性値を乗算

して得られる値と、ステップ１２で算出されたマージンに基づく受払金額属性値MarginCF__FVと、現TCF クラスの指数とは独立した受払金額属性値NonIndexCF__FV（図１６参照）とが加算され、その加算結果が現TCF クラスの受払総額属性値 TotalCF__FV（図１６参照）とされる。ここで、現TCF クラスのWeight属性のデフォルト値は１であるが、ユーザは、後述するユーザ・インタフェースであるマジック・シートを用いて、その値を個別に変更できる。また、現TCF クラスの指数とは独立した受払金額属性値NonIndexCF__FVは、後述するユーザ・インタフェースを介してユーザにより指定される、現TCF クラスが属するサイドのTCFChainクラスの初期元本交換額IniPrincAmntまたは終期元本交換額FinPrincAmntのいずれかよりコピーされたものである（図１９のステップ８または１０を参照）。

【０１５０】

最後に、ステップ１４においては、ステップ１３で算出された現TCF クラスの受払総額属性値 TotalCF__FVに、現TCF クラスのディスカウント・ファクター属性値DF(PaymDate)が乗算され、その乗算結果が現TCF クラスの受払総額現在価値 TotalCF__PV（図１６参照）とされる。現TCF クラスのディスカウント・ファクター属性値DF(PaymDate)は、現TCF クラスの生成時（図１９のステップ４，８，１０参照）に、現TCF クラスの受払日属性値PaymDateを引数として、現TCF クラスが属するサイドのTCFChainクラスが参照しているDiscCurve インスタンス（図１３参照）のGetFactor メソッドを呼び出すことにより、予め算出されているものであり、将来価値としての受払総額 TotalCF__FVを現在価値に割り引くためのファクターである。

【０１５１】

このようにして算出された現TCF クラスの受払総額現在価値 TotalCF__PVは、TCFChainクラスのUpdateCFs メソッドを介してTVirtualContractインスタンスまたはTCFSwap インスタンスにおいてGetNPV()メソッドの戻り値として、TVirtual Contractインスタンスまたは TCFSwap インスタンスに返されることになる。

【０１５２】

TVirtualContractインスタンスまたはTCFSwap インスタンスのGetNPV()メソッドでは、各サイド（レシーブサイドおよびペイサイド）ごと、すなわち、各TCFC

hainクラスごとに、それぞれに属するTCF クラスから返された受払総額現在価値TotalCF__PVを加算し、その加算結果を各サイドの現在価値PVと、さらに、各サイドの現在価値PVを加算して得られる総現在価値NPVとを、後述するユーザ・インタフェース上に表示する。

【０１５３】

以上説明したようにして、ユーザが後述するユーザ・インタフェースを介して任意のパラメータを変更した場合に、即座に新たな現在価値を算出し直すことができ、これにより効率的なリスク管理が実現される。

【０１５４】

キャッシュ・フロー系取引を実現するTCFSwapクラスの構造と生成手順について説明してきたが、キャッシュ・フロー系取引のインスタンス生成の実際を分類して、以下に説明する。

【０１５５】

既に説明した通り、TCFSwapクラスは、キャッシュ・フローを交換するほとんどの取引を生成できるが、実際の生成に当たっては対象とする取引を基本的な性質によって分類する必要がある。

【０１５６】

キャッシュ・フロー系取引は、その性質により、以下に示されるように分類される。

【０１５７】

1)異なる種類の元本交換

ここで「元本」は、TCFChainクラスに対して設定されるNonIndexCF#FV 、すなわち、指数に依存しないで決定されるキャッシュ・フローという意味で用いている。

【０１５８】

2)同じ種類の元本の時間軸上の交換

3)異なる種類の指数により生成されるキャッシュ・フローの交換

4)指数により生成されるキャッシュ・フローと元本の時間軸上の交換

上述の分類を、市場における代表的な取引に対応させたものが図２１の表である。表中、株式・商品現先は分類番号２と４に現れているが、これは、これらの取引がどちらの形態でも表現可能であることによる。取引に対する認識の仕方によっては、これら以外にも分類先が変わる場合もある。各取引の性質を損なわずに表現できるのであれば、必ずしもこの分類に従う必要はない。また、取引によっては、この分類の組合せとして構成されるものもある。通貨スワップなどは、これに該当する取引で、分類番号１と３の組合せとして表現できる。

４．　オプション系取引を実現するTOptionコンテナ・クラス

次に、本発明の第４の特徴であるTOption コンテナ・クラスの枠組みについて説明する。

４．１　TOption コンテナ・クラスの特徴

キャッシュ・フロー系取引が、単純なキャッシュ・フロー要素（CashFlowLet）の集合様式を多様化させることで実在する多くの取引を表現したのに対して、オプション系取引はメソッドを多様化することでこれを表現する。これは、キャッシュ・フロー系取引の価格評価（時価評価）が、統一された手法で定義できるのに対して、オプションでは、取引種ごとにその価格評価モデルが異なることに加え、同一取引種に対しても異なる評価モデルが存在することが背景となっている。

【０１５９】

各価格評価モデルに、異なる評価メソッドを実装することは、抽象オプション・クラス上に価格評価の仮想メソッドを用意し、実際の価格評価メソッドは、これを継承するオプションの実体クラスに実装することで容易に実現できる。

【０１６０】

本実施の形態において実現されるオプション・クラスの際立った特徴は、原資産の取り扱いである。オプションの実体クラスには、当該オプションの属性に加え、原資産を定義する属性を用意するのが一般的であると考えられる。この方式

では、例えば次の様なオプション・クラスを用意することになる。

【０１６１】

1)為替用ブラック・ショールズ・モデル・クラス

2)株式用ブラック・ショールズ・モデル・クラス

3)商品用ブラック・ショールズ・モデル・クラス

つまり、「原資産＋評価モデル」で１つのオプション・クラスを定義するわけである。これに対して、本実施の形態では、原資産属性をリンクリストへの参照として抽象オプション・クラスに実装する。すなわち、抽象オプション・クラスは、a)不特定の原資産インスタンスを、b)任意数格納するコンテナとしての役割を主に担うのである。

４．２　TOption コンテナ・クラスのデータ構造

TOption は、抽象契約クラスTContract （図５、図８参照）を継承する抽象クラスで、その構造とインスタンス・イメージは図２２に、各属性とメソッドの概要は図２３に示す通りである。

【０１６２】

図２２において、TOption インスタンスは、原資産クラスであるTContract クラスの集合を格納するリスト構造への参照である、Underlyingという名前のリンクリストを保持する。

【０１６３】

また、TOption インスタンスは、それぞれ、TPwrGadgetクラスからそれぞれ継承される、DiscCurve という名前のTCntYieldCurveクラスのインスタンス、およびVolaCurve という名前のTCntVolaCurve クラスのインスタンスへの参照を保持する。

【０１６４】

この図・表から明らかなように、TOption インスタンスは、オプションとして本質的に必要となる属性と、不特定・任意数の原資産インスタンス（TContract

）を保持し、原資産の可変パラメータのうち、どれをストライクと比較するかを特定後、原資産インスタンスに対して必要な計算の実行を指示する。ここで、可変パラメータとは、金利・収益率・価格等の想定元本に乗ぜられる指数や、元本の交換レート、NPV 等、オプションのストライクと比較可能なレートを意味している。

４．３　TOption コンテナ・クラスの利用の優位性

TOption コンテナ・クラスを利用することの優位性は、以下の２点である。

４．３．１　コード・サイズとクラス数の縮小化

４．３．２　シミュレーションの高速化

以下に、上記各優位性について、順次説明する。

４．３．１　コード・サイズとクラス数の縮小化

TOption インスタンスは、実行時に任意の原資産インスタンスを格納できるため、格納されたインスタンスに存在するストライクと比較可能なパラメータのうち、どれを比較対象とするかを決定する処理が必要となる。これが決定されるとTOption インスタンスは、原資産クラスのインスタンスに実装されるアット・ザ・マネー（以下、ATM ）レベルを計算するGetATMメソッドを呼び出す。この戻り値が、TOption インスタンスのGetNPVメソッドのパラメータに渡され価格評価メソッドの実行は完了する。

【０１６５】

このように、TOption クラスは、それが保持する原資産インスタンスにATM レベルを提供するよう指示するのみで、実際のATM 演算は原資産インスタンスが実行する。ここで重要なのは、ATM レベルの算出は、オプションの原資産となる場合にのみ必要とされるのではなく、それが単体で（オプション・インスタンスに保有されないで）存在する場合でもプライシング等の実行時に必要となる機能であるという事実である。

【０１６６】

つまり、TOption インスタンスは、原資産インスタンスに予め備わっているメソッドを利用することで、自身の価格評価メソッドの実装を省力化しているのである。

【０１６７】

また、ATM レベルを取得する手続きがTOption に実装されることにより、ここから継承して実装されるオプションの実体クラスは、主にその価格評価モデルに固有の評価式のみを実装することで足りることになる。つまり、これらの実体クラスは、メモリ上で各価格評価メソッドが格納されるコード領域への参照を提供するために存在していると考えることもできる。同様の背景で、４．１項の1)～3)に上げたような分類は必要なく、Black-Sholesクラスのみを用意することで足り、クラス数を縮小する効果も得られる。

４．３．２　シミュレーションの高速化

評価日付を将来のある時点に設定して、その時点におけるオプション価値を計算することは、オプションのリスク分析として一般的に行われる行為である。本実施の形態では、TOption インスタンスの生成時に原資産インスタンスをも生成・保持する方式が採用されているため、このような将来の価格シミュレーション実行時にも良好なパフォーマンスを提供することができる。

【０１６８】

このようなシミュレーションで問題となるのは、設定された将来日付が、既に当該オプションのイン・ザ・マネーになった以降である場合である。その場合には、オプションとして価格変動分析を行うのではなく、権利行使されたことを想定し、そこで発生する原資産に対する価格分析を実行しなければならない。

【０１６９】

この際、本実施の形態が採用する方式では、予め原資産インスタンスが生成・保持されているため、イン・ザ・マネーになった段階で当該TOption インスタンスが原資産インスタンスを顕在化させるだけで目的を達することができる。つまり、アウト・オブ・ザ・マネー時には、原資産インスタンスは存在するものの、

同インスタンスの持つ時価は無視されTOption インスタンスの時価で置き換えられ、イン・ザ・マネーになった段階で、今度はTOption インスタンスの時価が原資産インスタンスの時価に取って代わられる処理が実装されるのである。

【０１７０】

この方法を採用しない場合には、オプションがイン・ザ・マネーになった段階で原資産を生成する手続きを実行する必要があり、演算処理としては負荷の重いものにならざるを得ない。この点が、本実施の形態の方式の優位性である。

５．金融カーブ定義機能

前述したように、TCFSwap インスタンスに属するTCFChainインスタンスまたはTOption インスタンスは、TPwrGadgetクラスから継承される各カーブ・インスタンスへの参照を保持し、これらが提供する金融カーブ特性を利用する（図１３および図２２参照）。

【０１７１】

TPwrGadgetクラスは、イールド・カーブ（TCntYieldCurveクラス）、プライス・カーブ（TCntPriceCurveクラス）、ボラティリティ・カーブ（TCntVolaCurveクラス）、通貨・商品・証券等の取引単位（TUnit クラス）、および休日除外対象都市（TCenter クラス）に関する情報を任意に定義・生成・保持するクラスである。

【０１７２】

図１３で説明したように、CashFlowLet であるTCF クラスのインスタンスの集合を格納するTCFChainコンテナ・クラスは、DiscCurve （ディスカウント・カーブ）にTCntYieldCurveインスタンス、PriceCurve（プライス・カーブ）にTCntPriceCurveインスタンスへの参照を保持し、TCntYieldCurve、およびTCntPriceCurveに実装されているGetFactor メソッドにより、それぞれ要求日付におけるディスカウント・ファクターと価格指数を取得する。

【０１７３】

また、図２２で説明したように、TOption クラスはDiscCurve （ディスカウント・カーブ）にTCntYieldCurveインスタンス、VolaCurve （ボラティリティ・カ

ーブ）にTCntVolaCurve インスタンスへの参照を保持し、GetFactor メソッドとGetVola メソッドにより、それぞれディスカウント・ファクターとボラティリティを取得する。

【０１７４】

また、本実施の形態におけるTPwrGadgetクラスでは、カーブ・インスタンスを任意数保持できるコンテナ・クラスTCntVirtualCurveで表現する「仮想カーブ」という手法を採用している。仮想カーブに計算メソッドを持たせることにより、コンテナに保持される各カーブ・インスタンスのGetFactor メソッドで取得したディスカウント・ファクターまたは価格指数に対して、特定の演算を実行することができる（例えば価格指数の平均など）。このように、本実施の形態においては、複数のカーブ・インスタンスを合成する機能も提供することができる。

【０１７５】

そのほか、特には図示しないが、本実施の形態では、イールド・カーブ（TCntYieldCurve）とFutures （TCntFuturesCurve）や、プライス・カーブ（TCntPriceCurve）とアドオン（TCntAddOn ）をそれぞれ合成する機能も付加することができる。

６．ユーザ・インタフェース

６．１　ユーザ・インタフェースの全体説明

本実施の形態では、リスク管理のためのパラメータ変更とそれに対するシミュレーション結果の表示を容易に行うことのできるユーザ・インタフェースが提供される。

【０１７６】

この場合に、1)Cash Flows,2)Options,3)Listed という３つの分類で入力フォームを提供するという新しい概念が採用されている。例えば、従来のシステムでは、スポット為替取引、為替スワップ取引、金利スワップ取引、・・・といった取引種別ごとに入力フォーム（または画面）が提供されていたのに対して、本実施の形態では、これらの取引が"Cash Flows"取引用の１つのインタフェースから

入力される。

【０１７７】

金融取引の複雑化に比例してオペレーションが困難になる一般的な統合管理システムとは異なり、このシステムの利用者には、目的取引のカテゴリーを指定するだけで、該当するフォームにアクセスすることができるという格段の利便性が提供される。

６．２　キャッシュ・フロー取引の入力手順の概要

まず以下の説明において、「左クリック」または「右クリック」とは、マウスの左ボタンまたは右ボタンを１回クリックさせる操作を意味する。「左ダブルクリック」とは、マウスの左ボタンを素早く２回クリックさせる操作を意味する。また、「ＡからＢまでマウスをドラッグアンドドロップさせる」とは、Ａの位置でマウスの左ボタンを押し下げ、そのままＢまでマウスカーソルを移動させ、Ｂの位置で押し下げていたマウスの左ボタンから指を離す操作を意味する。

【０１７８】

図２５は、本実施の形態のメインのコントロールパネルである。

【０１７９】

図２６は、キャッシュ・フロー取引を作成するための初期ウインドウ画面であり、図５７は、各部分が明確になるように強調した図である。

【０１８０】

まず、各タブエリアについて説明する。

【０１８１】

"Primary" タブエリアは、取引種規定と日付生成のためのパラメータを指定するエリアである。

【０１８２】

"Commodity/Discount"タブエリアは、受払通貨（あるいは商品単位）とディスカウント・カーブを指定するエリアである。

【０１８３】

"Principal Cash Flows"タブエリアは、想定元本に関連しない、直接交換され

る元本額を指定するエリアで、為替、為替スワップや通貨スワップの初期、終期交換元本を入力する場合に利用する。

【０１８４】

"Return Properties (1)" タブエリアは、想定元本に金利、収益率、価格の何れかの指数を乗ずることで生成されるキャッシュ・フローの基本条件を指定するエリアである。本実施の形態では、このキャッシュ・フローを、"IndexCF" と呼ぶ。

【０１８５】

"Return Properties (2)" タブエリアは、"Return Properties (1)" タブエリアで対応できないキャッシュ・フローを扱うための拡張条件を指定するエリアであり、非日常的な受払日、値決日を生成させる場合等に利用される。

【０１８６】

"Centers" タブエリアは、受払日と値決日の決定時に参照される休日除外対象都市を、レシーブサイドとペイサイドとで独立して指定するエリアである。

【０１８７】

次に、上記各タブエリア内の各アイコン、各ボタンまたは各フィールドにつき説明する。

【０１８８】

"Primary" タブエリア内のヘルメットアイコンから同エリア内のゴミ箱アイコンまで、マウスをドラッグアンドドロップさせることにより、現在作成しているオブジェクトを破棄することができる。

【０１８９】

"Primary" タブエリア内のヘルメットアイコンを左ダブルクリックすることにより、取引オブジェクトが構築、保持される。

【０１９０】

"Primary" タブエリア内の特には図示しない握手アイコン（ヘルメットアイコンから変化したもの）から虫眼鏡アイコンまで、マウスをドラッグアンドドロップさせることにより、取引オブジェクトの詳細を示すマジック・シートを表示させることができる。

【０１９１】

"Primary" タブエリアの"Kind"エリア内のリストフィールドを左クリックすることで、取引種定義の一覧が表示され、定義済みの取引種を選択することができる。取引種を選択することにより、フォーム上の入力必須フィールドがハイライト表示され、各フィールドには初期値が代入される。利用者は、このようにして呼び出された取引種定義に対して、自由に追加・修正を行うことができる。

【０１９２】

"Primary" タブエリア内の"Change Side" ボタンを左クリックすることで、後述するレシーブサイドとペイサイドに指定されたパラメータ群を、両サイド間で一括して入れ替えることができる。例えば、フォーム上に“ドル売り／円買い”のパラメータ群が指定されているときに、このボタンがクリックされると、“ドル買い／円売り”のパラメータ設定状態に切り替わる。

【０１９３】

"Primary" タブエリアの"Data Generation" エリア内の"Business Day Conv" フィールドでは、休日除外方式を指定することができ、"Modified"、"following"、"Preceding"、または"No Adust"を指定できる。ここで設定された値は、レシーブサイドおよびペイサイドの各TCFChainクラスのBDConv属性値（図１４参照）として設定される。

【０１９４】

"Primary" タブエリアの"Data Generation" エリア内の"Centers" フィールドには、休日除外対象都市を表示することができる。これらの都市情報は、後述する"PwrGadget" 機能によって予め定義しておくことができる。ここの設定値は、レシーブサイドおよびペイサイドの各TCFChainクラスのPaymCenters 属性値（図１４参照）として設定される。

【０１９５】

"Primary" タブエリアの"Data Generation" エリア内の"Today" フィールドには、評価基準日（取引日／本日）を指定することができる。

【０１９６】

"Primary" タブエリアの"Data Generation" エリア内の"Spot"フィールドには

、スポット日（Spot Date/Settlement Date ）を指定することができる。

【０１９７】

"Primary" タブエリアの"Data Generation" エリア内の"Effective" フィールドには、実効日（Effective Date）を指定することができる。ここで設定された値は、レシーブサイドおよびペイサイドの各TCFChainクラスのEDate 属性値（図１４参照）として設定される。

【０１９８】

"Primary" タブエリアの"Data Generation" エリア内の"Stub"フィールドには、スタブ指定時の次回受け払い日を指定することができる。ここで設定された値は、レシーブサイドおよびペイサイドの各TCFChainクラスのFDate 属性値（図１４参照）として設定される。

【０１９９】

"Primary" タブエリアの"Data Generation" エリア内の"Maturity"フィールドには、満期日（Termination Date）を指定することができる。ここで設定された値は、レシーブサイドおよびペイサイドの各TCFChainクラスのTDate 属性値（図１４参照）として設定される。

【０２００】

"Primary" タブエリアの"Data Generation" エリア内の"EndEnd"チェックボックスでは、月末ロール指定を指定することができる。ここで設定された値は、レシーブサイドおよびペイサイドの各TCFChainクラスのEndEnd属性値（図１４参照）として設定される。

【０２０１】

"Primary" タブエリアの"Data Generation" エリア内の"AdjMty"チェックボックスでは、満期日シフト指定を指定することができる。ここで設定された値は、レシーブサイドおよびペイサイドの各TCFChainクラスのAdjTDate属性値（図１４参照）として設定される。

【０２０２】

続いて、"Commodity/Discount"タブエリア内の"Rec(+)"エリアおよび"Pay(-)"エリア内の各"Commodity" フィールドでは、受払通貨または商品単位を指定する

ことができる。これらの通貨または単位は、後述する金融特性の定義機能（"Pwr Gadget" 機能）によって予め定義しておくことができる。ここで設定された値は、"Rec(+)"エリアまたは"Pay(-)"エリアに対応するレシーブサイドまたはペイサイド（以下、対象サイドと呼ぶ）に対応するTCFChainクラスのAppliedUnit 属性値（図１４参照）として設定される。

【０２０３】

"Commodity/Discount"タブエリア内の"Rec(+)"エリアおよび"Pay(-)"エリア内の各"DiscCurve" フィールドでは、レシーブサイドまたはペイサイドのそれぞれにおけるキャッシュ・フロー群を割り引くためのイールド・カーブを指定することができる。また、対象取引がスポット為替である場合であって、"Primary" タブエリアの"Data Generation" エリア内の"Today" フィールドに設定される評価基準日が本日である場合には、上記エリア内の"Stub"フィールドに設定されるスポット日から本日まで割り引くためのカーブを指定することができる。これらのカーブは、後述する金融特性の定義機能（"PwrGadget" 機能）によって予め定義しておくことができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのDiscCurve 属性値（図１４参照）として設定される。

【０２０４】

次に、"Principal Cash Flows"タブエリア内の中央の矢印ゲージは、元本交換時の基準通貨サイドを規定することができる。例えば、Rec:"USD"/Pay:"JPY" （ドル買い／円売り）が指定されている場合に、基準サイドを"Rec" 側に指定すれば、「1USD=130.05JPY」のようにドル基準が採用され、逆に基準サイドを"Pay" 側に指定すれば、「1JPY=0.0076894USD 」のように円基準が採用される。

【０２０５】

上記矢印ゲージの下の第１行目のフィールドには、初期元本交換レートが表示され、第２行目のフィールドには、終期元本交換レートが表示される。上記ゲージ指定とこれらのフィールドに表示されるレートとに基づいて、基準通貨に対する他方の通貨額が自動的に演算される。

【０２０６】

"Principal Cash Flows"タブエリアの"Initial: Pay(-)/Final: Rec(+)" エリ

アおよび"Initial: Rec(+)/Final: Pay(-)" エリア内の各第１行目のフィールドには、為替スワップおよび通貨スワップに関して、"Primary" タブエリアの"Data Generation" エリア内の"Effective" フィールドに設定された実効日に交換される交換元本実額（初期元本交換額）を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのIniPrincAmnt属性値（図１４参照）として設定される。

【０２０７】

"Principal Cash Flows"タブエリアの"Initial: Pay(-)/Final: Rec(+)" エリアおよび"Initial: Rec(+)/Final: Pay(-)" エリア内の各第２行目のフィールドには、為替スワップおよび通貨スワップに関して、"Primary" タブエリアの"Data Generation" エリア内の"Maturity"フィールドに設定された満期日に交換される交換元本実額（終期元本交換額）を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのFinPrincAmnt属性値（図１４参照）として設定される。

【０２０８】

次に、"Return Properties (1)" タブエリア内の中央の矢印ゲージは、クーポン・スワップ等を入力する場合の想定元本変換時の基準サイドを規定することができる。その機能は、"Principal Cash Flows"タブエリア内の中央の矢印ゲージの場合と同様である。

【０２０９】

上記矢印ゲージの下のフィールドには、想定元本交換レートが表示される。上記ゲージ指定とこれらのフィールドに表示されるレートとに基づいて、基準通貨に対する他方の通貨額が自動的に演算される。

【０２１０】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"Notional"ラジオボタンでは、固定元本制（"Fixed":"Fixed Notional Principal"）または可変元本制（"Var":"Variable Notional Principal" ）の何れかの想定元本定義を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのVarNotional 属性値（図１４参照）として設定され

る。

【０２１１】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"Notional"フィールドには、プレーンバニラ取引の場合にはその想定元本額、アモチ等の想定元本が一定でない取引の場合には初期想定元本を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのNotionalAmnt属性値（図１４参照）として設定される。

【０２１２】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"IndexType" ラジオボタンでは、金利（"Int"）、収益率（"%chg"）、または価格（"Prx"）の何れかの指数を定義することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのIndexType 属性値（図１４参照）として設定される。

【０２１３】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"Index/Value" フィールドには、指数値とその単位（"%","bp","dec"）を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのIndexVal属性値およびIndexUnit 属性値（図１４参照）として設定される。

【０２１４】

上記各"Index/Value" フィールドの左側の各ゲージでは、固定（"FX"）または変動（"FL"）の指数定義を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのFloatIndex属性値（図１４参照）として設定される。

【０２１５】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"ReturnCurve" フィールドでは、レシーブサイドまたはペイサイドのそれぞれにおけるフォアード・プライス・カーブまたはイールド・カーブを指定することができる。フォアード・プライス・カーブは、"Forward Forex","Equity Sw

aps","Commodity Swaps"等の取引が定義される場合に指定される。イールド・カーブは、"Interest Rate Swaps" 等の取引が定義される場合に指定される。これらのカーブは、後述する金融特性の定義機能（"PwrGadget" 機能）によって予め定義しておくことができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのPriceCurve属性値（図１４参照）として設定される。

【０２１６】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"D-Cnt + C/R-Frq" フィールド群では、日割り計算方式（"Act/360","Act/365Fixed","Act/Act","30/360","30E/360" ）、受払頻度（支払い間隔）、およびリセット頻度を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのDayCount属性値、CFFrq 属性値、およびResetFrq属性値（図１４参照）として設定される。

【０２１７】

"Round/Trunc" フィールド群では、四捨五入（"Rnd" ）／切捨て（"Trc" ）、およびその実効桁を指定することができる。ここで設定された値は、対象サイドに対応するTCFChainクラスのRounding属性値およびDecPlaces 属性値（図１４参照）として設定される。

【０２１８】

図２７は、必要なパラメータをアクティベートする図である。まず画面左下の鍵アイコンを左クリックした後、所望のパラメータ上で右クリックして"Active" プロパティを左クリックしてチェックする。金利スワップの場合は、かなりの箇所がアクティブにされる。

６．３　金利スワップ取引の作成手順

ここでは、円と円のキャッシュ・フローの交換の金利スワップ、いわゆる円／円スワップを作成する。

【０２１９】

まず、画面上半分の"Commodity/Discount"タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"Commodity" フィールドにおいて通貨を選択する。

【０２２０】

次に、上記各エリア内の"DiscCurve" フィールドにおいて、上記各選択した通貨に対応するイールド・カーブ特性として、"LIBOR&SWAP"を指定する。

【０２２１】

続いて、画面下半分の"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"ReturnCurve" フィールドにおいて、リターン・カーブ特性として"LIBOR&SWAP"を指定する。このリターン・カーブは、フォワードのキャッシュ・フロー、すなわち将来の金利を計算する基礎になる。

【０２２２】

"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリア内の各"D-Cnt + C/R-Frq" フィールド群には、マーケットのコンベンションに従った各値を設定する。

【０２２３】

このようにして作成したパラメータセットは、図２８に示されるように、画面左下の鍵アイコンの右のアイコンを左クリックして表示されるダイアログボックスを用いて、名前を付与することができる。ここでは、例えば"JPY/JPY#SWP" という名前を付与する。この名前は、画面左上の"Primary" タブエリアの"Kind"エリア内のリストフィールドに登録され、以後その名前を選択することにより、それに対応するパラメータセットを呼び出すことができる。

【０２２４】

ここで、４年ものの金利スワップ商品を作成する。まず、図２９に示されるように、画面左の"Data Generation" エリア内の"Today" フィールド、"Spot"フィールド、"Effective" フィールド、および"Maturity"フィールドを順次設定する。

【０２２５】

次に、図３０に示されるように、画面下半分の"Return Properties (1)" タブエリアの"Rec(+)"エリアおよび"Pay(-)"エリアの各"Notional"フィールドに、１０億日本円を設定する。

【０２２６】

また、フィックスレート（固定金利）を受けて（"Return Properties (1)" タブエリアの"Rec(+)"エリアにおいて"FX"を選択）、フロートを払う（同じく"Pay (-)"エリアにおいて"FL"を選択）という金利スワップを設定する。この場合さらに、"Return Properties (1)" タブエリアの"Rec(+)"エリア内の"Index/Value" フィールドに、フィックスレートとして例えば1.75527%をプライシングする。この意味は、将来４年間フロートの"LIBOR" レートを払ってそれに対して1.75527%の固定金利を受ければ、この取引において、払ったお金のＰＢと受けるお金のＰＢが等しくなるということになる。

【０２２７】

その後、図３０に示される画面左上の"Primary" タブエリア内のヘルメットアイコンを左ダブルクリックすることによって、そのアイコンが図３１に示されるように握手アイコンに変化し、上述のようにして定義されたスワップに対するキャッシュ・フローオブジェクトが構築される。ここでは、前述した２．３項の、TCFSwap/TCFChain/TCFインスタンスの生成手順が実行される。

【０２２８】

キャッシュ・フローオブジェクトの詳細は、図３１に示される画面左上の"Primary" タブエリア内において、握手アイコンから虫眼鏡アイコンまでマウスをドラッグアンドドロップさせることにより、図３２～図３４に示されるように表示されるマジック・シートを介して確認および変更することができる。"REC" タブと"PAY" タブのそれぞのエリアでは、レシーブサイド（受取り側）およびペイサイド（支払い側）のそれぞれに関する指定期間内（ここでは４年間）の各キャッシュ・フロー期間のそれぞれについて、その詳細を確認および変更することができる。

【０２２９】

ここに表示される各行は、レシーブサイドおよびペイサイドのそれぞれに対応するTCFChainインスタンスから参照される各TCF インスタンスに対応し、各行の各フィールドは、各TCF インスタンスが保有する各属性値（図１６参照）に対応している。第１行めに手のマークが付加されていないフィールドは、その値を変更し新たなプライシングを行うことができる。例えば、あるキャッシュ・フロー

期間における"NotionalAmnt"フィールド（"Notional"フィールドに等価）の値を変更し、"Update"アイコンを左クリックすることにより、新たなプライシングが実行される。この場合には、前述した３．項のTCF インスタンスの更新メソッドが実行されることになる。

【０２３０】

図３２～図３４では、レシーブサイドに関する"REC" タブエリアのマジック・シートが表示されている。図３５では、ペイサイドに関する"PAY" タブエリアのマジック・シートが表示されている。

６．４　エクイティ・スワップ取引の作成手順

次に、エクイティ・スワップ取引の作成を行う。図３６にその設定画面例を示す。ここでは、将来４年間にわたって、"ReturnCurve" フィールドに設定されている"Nikkei Forward"というリターン・カーブ特性に従うフロートレート（変動金利）を受けて（"Return Properties (1)" タブエリアの"Rec(+)"エリアにおいて"FL"を選択）、フロートの"LIBOR" レートを支払う（同じく"Pay(-)"エリアにおいて"FL"を選択）というスワップを設定する。この場合さらに、"Return Properties (1)" タブエリアの"Pay(-)"エリア内の"Index/Value" フィールドに、マージンレートとして例えば-1.10993% をプライシングすることができる。

【０２３１】

図３７は、上述のようにして作成されたエクイティ・スワップオブジェクトのマジック・シートである。ここで、"PrePrice"フィールドの各値は、リターン・カーブ特性"Nikkei Forward"によって算出されるものである（図２０のステップ８参照）。

６．５　為替取引の作成手順

次に、為替取引の作成を行う。図３８は、為替商品の作成画面である。ここでは、"Principal Cash Flows"タブエリアの、"Initial: Pay(-)/Final: Rec(+)" エリアの第２行目のフィールド（終期元本交換額フィールド）に設定されたＵＳドルを買う。"Principal Cash Flows"タブエリアの中央の無名のフィールドを左

ダブルクリックすると、現在の交換レートが表示される。このレートで良ければ、"Principal Cash Flows"タブエリアの"Initial: Rec(+)/Final: Pay(-)" エリアの第２行目のフィールド（終期元本交換額フィールド）に設定された日本円の額を支払うという取引になる。

【０２３２】

図３９および図４０は、上述のようにして作成された為替取引のマジック・シートである。データ構成はスワップの場合と全く同様であるが、為替取引の場合は、キャッシュ・フローは１件（１行）のみ存在する。為替取引の性格上、"NotionalAmnt"フィールドや、それに基づいて計算される"IndexCF#FV"フィールド等には値は存在しない。その代わり、図３９に示されるように、"REC" タブエリアの"NonIndexCF#FV" フィールドに、"Principal Cash Flows"タブエリアの"Initial: Pay(-)/Final: Rec(+)" エリアの第２行目のフィールド（終期元本交換額フィールド）に設定されたＵＳドルの値が設定されている。また図４０に示されるように、"PAY" タブエリアの"NonIndexCF#FV" フィールドに、"Principal Cash Flows"タブエリアの"Initial: Rec(+)/Final: Pay(-)" エリアの第２行目のフィールド（終期元本交換額フィールド）に設定された日本円のマイナス値が設定されている。そして、"NPV" フィールドの値は、レシーブサイドのPV値（図３９）とペイサイドゼロのPV値（図４０）がバランスして、０となる。

６．６　金融特性の定義手順（"PwrGadget" 機能）

続いて、各種金融特性の定義手順（"PwrGadget" 機能）について説明する。図４１は"PwrGadget" 機能の初期画面であり、"Curve Yard"タブエリアに、イールド・カーブを定義する"Yield" 項目、プライス・カーブを定義する"Price" 項目、フューチャーズ・カーブを定義する"Futures" 項目、ボラリティ・カーブを定義する"Vola"項目、仮想カーブを定義する"Virtual" 項目、取引単位を定義する"Unit"項目、休日除外対象都市を定義する"Center"項目が表示されている。

【０２３３】

"Curve Yard"タブエリアの"Center"項目を左ダブルクリックすると、図４２に示されるように、休日除外対象都市を定義する詳細項目が表示される。ここでは

、都市ごとに、休祭日を排除するための情報を定義することができる。

【０２３４】

"Curve Yard"タブエリアの"Unit"項目を左ダブルクリックすると、図４３に示されるように、取引単位を定義する詳細項目が表示される。ここでは、通貨の取引単位、特定の会社の株の単位、オイルの単位、債権等を、自由に定義することができる。

【０２３５】

"Curve Yard"タブエリアの"Yield" 項目を左ダブルクリックすると、図４４に示されるように、イールド・カーブを定義する詳細項目が表示される。図４４の"Rate on Grid"タブエリアには、"LIBOR&SWAP(JPY/JPY)" と名付けられたイールド・カーブを定義するための画面が表示されている。まず図４５の"Curve Yard"タブエリアに表示されているように、１ヶ月おきとか１年おきとかというグリッドを自由に定義することができ、この場合の各グリッドの属性は、その詳細については省略するが、図４５の"Linked Object" タブエリアにおいて定義することができる。そして、"Rate on Grid"タブエリアに表示されたグリッドごとに、"Rate"フィールドにレートを定義することができ、これによりカーブが定義されたことになる。

【０２３６】

"Rate"フィールドに対するレートは、手動による設定も可能であるが、例えばネットワークを介した定期的な取得（デジタルフィード）によって設定されるように構成することもできる。

【０２３７】

"Curve Yard"タブエリアの"Price" 項目を左ダブルクリックすると、図４６に示されるように、プライス・カーブを定義する詳細項目が表示される。ここでは例えば、日経インデックスというような、直接将来の価値が観測できるようなカーブを、図４４および図４５で示したイールド・カーブの場合と同様にして定義することができる。この場合、図４６の"Rate on Grid"タブエリアの"Rate"フィールドには、プライス（価格）が直接入力されることになる。

【０２３８】

図４７は、オイル等のコモディティ（商品）に対するプライス・カーブの定義画面である。コモディティにおいては、スポットの価値と将来の価値は関係がない。そこで、図４７の"Rate on Grid"タブエリアに示されるように、"spt" グリッドのみが定義される。

【０２３９】

"Curve Yard"タブエリアの"Virtual" 項目を左ダブルクリックすると、図４８に示されるように、仮想カーブを定義する詳細項目が表示される。ここでは、"Curve Yard"タブエリアの"Test"項目下に示されるように、イールド・カーブやプライス・カーブ等の他の複数のカーブをまとめてコンテナに登録し、それら複数のカーブ間の関数（演算メソッド）を定義することにより、新たな仮想的なカーブを定義することができる。例えば、図４８において、"Linked Object" タブエリア内の"CalcMethod"フィールドにおいて、カーブ間の演算を定義することができる。例えば"AverageMethod "メソッドは、複数のカーブの対応する各値の平均を演算するメソッドである。また、"SumMethod" メソッドは、複数のカーブの対応する各値の合計を演算するメソッドである。上述のようにして定義された仮想カーブにおいては、そのコンテナに登録されている元のカーブの構成レート等が例えば前述したデジタルフィード等により変化した場合には、その変化を自動的に仮想カーブにも反映させることが可能となる。このような仮想カーブを用いることにより、従来なかった全く新しい金融取引を定義することが可能となる。

６．７　オプション取引の作成手順

次に、オプション取引の１つである、スワップション、つまりスワップについてのオプションを取り扱う取引の作成手順について説明する。

【０２４０】

まず、原資産であるスワップを定義する。図４９は、その作成画面例である。この例は、２年後スタートの４年ものという将来スタートのスワップであり、"LIBOR" のフロートレートを受けてフィックスレートを支払うというスワップの例である。

【０２４１】

上述のようなスワップについてのオプションとは、スタート年月日において、契約者が、その時点での状況を判断することにより、そのスワップを履行するか否かを選択できる権利を有するような取引をいう。

【０２４２】

上述のようにしてスワップについての定義を行った後に、前述のように、図４９に示される画面左上の"Primary" タブエリア内のヘルメットアイコンを左ダブルクリックすることにより、そのスワップに関するキャッシュ・フロー取引がまず構築され、上記ヘルメットアイコンが握手アイコンに変化する。

【０２４３】

続いて、メインのコントロールパネルの"Opt" アイコンを左クリックすることにより、図５０に示されるように、オプションの定義ウインドウが表示される。そして、スワップの定義ウインドウ上の"Primary" タブエリア内の握手アイコンからオプションの定義ウインドウの"Primary" タブエリア内の人アイコンまでマウスをドラッグアンドドロップさせると、オプションの定義ウインドウの画面表示が、図５１に示されるように変化する。この結果、TOption インスタンスが、上述のようにして定義されたスワップオブジェクトのインスタンスをコンテナに保有する結果となる。

【０２４４】

オプションの定義ウインドウにおいて各オプションに対応したパラメータ設定を行った後、そのウインドウの"Primary" タブエリア内のヘルメットアイコンを左ダブルクリックすることにより、そのアイコンが図５２に示されるように握手アイコンに変化し、定義されたオプションに対するTOption インスタンスが構築される。この場合特に、オプションのモデルごとにクラスが作られており、途中のパラメータ選択によりモデルが特定されると、そのクラスに必要なパラメータの設定フォームが自動的に表示される。

６．８　金融取引の合成手順

以上示したようにして個々に作成される金融取引は、任意に合成することができ、仮想取引を形成することができる。

【０２４５】

今、スワップの定義ウインドウやオプションの定義ウインドウで各オブジェクトが構築されて"Primary" タブエリア内のヘルメットアイコンが握手アイコンに変化した後、メインのコントロールパネルの"Opt" アイコンを左クリックすることによって、図５３に示されるように、ブックマネージャ（BOOK MANAGER）ウインドウが表示される。

【０２４６】

そして、スワップまたはオプションの定義ウインドウ上の"Primary" タブエリア内の握手アイコンからブックマネージャウインドウのフラスコアイコンまでマウスをドラッグアンドドロップさせる操作を複数の構築されたオブジェクトに対して行うと、それらのオブジェクトを１つの仮想取引としてとりまとめることができる。

【０２４７】

さらに、このようにしてフラスコアイコンにおいて１つの金融取引にとりまとめられた複合取引である仮想取引を、ブックマネージャウインドウ上で、任意に名前付けされたツリー構造としてデータベース管理することができる。そのためには、フラスコアイコンからブックマネージャウインドウのツリー構造上の任意のツリー要素までマウスをドラッグアンドドロップさせればよい。この場合に、図５４に示されるように表示される取引入力（DEAL ENTRY）ウインドウを使って、複合取引である金融取引に、契約先や取引種名等の属性を設定することができる。

【０２４８】

この結果、例えば図５５に示されるようにして、ツリー構造内にID(15)として示される複合取引である仮想取引が保存される。これが図２等に示されるTVirtualContractを構成する。

【０２４９】

ここで、上記ツリー構造内の金融取引を左ダブルクリックすると、図５６に示されるように、２枚のマジック・シートが表示される。これにより、この金融取引が、２つの取引実体から構成される複合取引であることがわかる。

７．本実施の形態を実現するプログラムが記録された記録媒体についての補足

本発明は、コンピュータにより使用されたときに、上述の本発明の実施の形態の各構成によって実現される機能と同様の機能をコンピュータに行わせるためのコンピュータ読出し可能記録媒体として構成することもできる。

【０２５０】

この場合、例えばフロッピィディスク、ＣＤ－ＲＯＭディスク、光ディスク等の可搬型記録媒体や、ネットワーク回線経由で、本発明の実施の形態の各種機能を実現するプログラムが、コンピュータ本体内のメモリにロードされて、実行される。

【０２５１】

図５の構成では、オンメモリ上の各クラスインスタンスを実行するクラスモジュール５０１、補助記憶装置上にリレーショナルデータベースを構築するＲＤＢモジュール５０３、およびクラスモジュール５０１とＲＤＢモジュール５０３との間でデータの中継を管理するＲＤＢＭＳモジュール５０２等が、コンピュータのハードウエア機能として実装される。ＲＤＢＭＳモジュール５０２は、ＳＱＬの発行を制御するTQueryメソッド、ストアドプロシジャを制御するTStored-Procメソッド等を実装する。

【０２５２】

【発明の効果】

本発明の第１の態様の構成によれば、１つ以上の取引実体から構成される金融に関する複合取引（複合商品）に対するリスク管理において、従来同時に達成することが困難であった、システムの単純化・リスク管理演算の高速化・データ圧縮という３つの要請を同時に達成することが可能となる。

【０２５３】

すなわち、本発明の第１の態様の構成では、対象となる取引の種類は問わずかつその構成要素が単一か複合かを問わずに、全ての取引を例外なく１つの仮想取引手段によって取りまとめることができる。このため、アプリケーション・プログラマは、取引種や構成を意識することなく、プログラミングを行うことができ

る。具体的には、取引実体手段がTContract クラスのインスタンスなどとして実装され、仮想取引手段がTVirtualContractクラスのインスタンスなどとして実装されることにより、全ての取引を常にTVirtualContractインスタンスなどとして認識することができる。

【０２５４】

そして、アプリケーション・プログラマは、TContract クラスなどの構造は変更することなく、仮想取引手段における複合取引の特性を算出する機能を実現するTVirtualContractインスタンスなどにおけるメソッドを個別に開発するだけで、様々なリスク管理機能を容易に開発することが可能となる。

【０２５５】

また、本発明の第１の態様の構成によれば、リスク管理における各種演算を、参照データ群を介した各取引実体手段に対するシーケンシャルなアクセス動作によって高速に実行することが可能となる。

【０２５６】

具体的には、取引実体手段を、時価評価演算機能をGetNPVメソッドとして保持するTContract クラスのインスタンスなどとして実装し、仮想取引手段を、参照データ群をリンクリストとして保持すると共に、複合取引の特性を算出する機能を仮想メソッドGetNPVとして保持するTVirtualContractコンテナ・クラスのインスタンスなどとして実装することにより、オンメモリ上での高速リスク管理演算が容易に実現される。

【０２５７】

さらに、本発明の第１の態様の構成によれば、１つの取引実体手段のデータを、複数の金融取引のために使い回すことが可能となる。この結果、同様の金融取引を大量に取り扱うような金融機関などにおいて、大きなデータ圧縮効果を得ることが可能となり、コンピュータ資源の削減が可能となる。

【０２５８】

この場合に、例えば、仮想取引手段内の複合取引の特性を算出する機能に、各演算結果に所定の変換係数を乗じて新たな演算結果を算出するような機能を持たせることにより、金融取引ごとに各取引実体手段に対する金融特性の微妙な差異

を吸収することが可能となる。

【０２５９】

本発明の第２の態様の構成によれば、為替、現金貸借、債権、株式、商品、金利・通貨スワップ、エクイティ・スワップ、コモディティ・スワップをはじめとする「交換」をベースとする多様な金融取引を、１つの取引系列モデル化手段によって統一的にモデル化することが可能となる。より具体的には、キャッシュ・フロー系取引実体が、受け側および払い側のそれぞれにおける単位取引期間ごとのキャッシュ・フロー要素（CashFlowLet ）の集合として管理され、各要素がそれらを参照する取引系列モデル化手段によって取りまとめられる構成を有する。そして、この参照様式が多様化されることによって、各種の「交換ベースのキャッシュ・フロー系取引」を統一的にモデル化することが可能となるのである。

【０２６０】

この結果、金融取引モデル化装置のシステム構成を非常にコンパクトなものにすることが可能となり、開発・販売コストの削減に大きく貢献する。

【０２６１】

ここで金融取引対象は、必ずしも通貨である必要はなく、為替、債権、株式、商品などであってもよい。この場合には、単位取引モデル化手段における時価評価演算機能は、それに対応する単位取引期間において、金融取引対象の単位（例えば株数など）のもとでの、金利、収益率、または価格に準ずる時価比率を演算することにより、将来価値指数を算出することができる。すなわち、本発明の第２の態様では、どのような単位を有する金融取引対象であっても、「交換ベースのキャッシュ・フロー系取引」を統一的に扱えるのである。

【０２６２】

また、本発明の第２の実施の形態の構成によれば、ユーザは、上述した「交換」をベースとする多様な金融取引を、統一したユーザ・インタフェースを介して設計することが可能となり、操作性の向上に大きく貢献する。

【０２６３】

さらに、本発明の第２の実施の形態の構成によれば、ユーザは、交換ベースのキャッシュ・フロー系取引に対し、単位取引期間ごとの非常に詳細なカスタマイ

ズを行うことが可能となる。

【０２６４】

本発明の第３の態様の構成によれば、原資産モデル化手段は、オプションの評価モデルとからは完全に分離・独立して設計・実装することができ、オプションの本体であるオプションモデル化手段は、参照データ群を介して上述のようにして独立して実装される任意数の原資産モデル化手段を参照することができる。すなわち、オプションモデル化手段に対して１つの指示を発行するだけで、参照データ群を介して各原資産モデル化手段内の時価評価演算機能が独立に時価評価演算を実行し、最後にオプションモデル化手段内のオプション取引の特性を算出する機能が、各演算結果と所定の評価モデル（イン・ザ・マネー時）または所定の評価モデルのみ（アウト・オブ・ザ・マネー）に基づいて、オプション取引の特性を算出することができる。このため、本発明の第３の態様の構成によれば、オプション取引のシステム設計が簡略化され、性能の向上と開発・販売コストの削減に貢献する。

【０２６５】

また本発明の第３の態様の構成においては、１つの原資産モデル化手段のデータを、複数のオプション取引のために使い回すことが可能となる。この結果、同様のオプション取引を大量に取り扱うような金融機関などにおいて、大きなデータ圧縮効果を得ることが可能となり、コンピュータ資源の削減が可能となる。

【０２６６】

さらに、本発明の第３の態様の構成においては、イン・ザ・マネー／アウト・オブ・ザ・マネーの別による切替え制御が単純化され、高性能なシステムを構築することが可能となる。

【０２６７】

本発明の第４の態様の構成によれば、ユーザは、金融取引対象の単位、休日除外都市、イールド・カーブ、プライス・カーブ、ボラリティ・カーブ等の各種金融特性を自由に時価評価演算に導入し、設計することも可能となる。

【０２６８】

また、本発明の第４の態様の構成によれば、ユーザは、複数の金融特性を合成

してさらに効果的な金融特性を時価評価演算に導入することが可能となる。

【図面の簡単な説明】

【図１】

本発明のブロック図（その１）である。

【図２】

本発明のブロック図（その２）である。

【図３】

本発明のブロック図（その３）である。

【図４】

本発明の概念図である。

【図５】

本発明の実施の形態の全体構成図である。

【図６】

解約権付きスワップの仮想取引管理の説明図である。

【図７】

プレーンバニラ・スワップの仮想取引管理の説明図である。

【図８】

仮想取引を実現するクラス構造図である。

【図９】

データ圧縮の説明図（その１）である。

【図１０】

データ圧縮の説明図（その２）である。

【図１１】

ＲＤＢＭＳ上での仮想取引の実現構造図である。

【図１２】

ＲＤＢ上のテーブル構成図（その１）である。

【図１３】

TCFSwap インスタンス・イメージとTCF クラスのデータ構造図である。

【図１４】

TCFChainクラスの属性概要を示す図である。

【図１５】

ＲＤＢ上のテーブル構成図（その２）である。

【図１６】

TCF クラスの属性概要を示す図である。

【図１７】

ＲＤＢ上のテーブル構成図（その３）である。

【図１８】

TCFSwap インスタンスの生成の手順の説明図である。

【図１９】

TCFChain.BuildCFメソッドの処理フローである。

【図２０】

TCF.Updateメソッドの処理フローである。

【図２１】

取引の分類を示す図表である。

【図２２】

TOption クラスのデータ構造とインスタンス・イメージを示す図である。

【図２３】

TOption の属性とメソッドの概要を示す図表である。

【図２４】

TPwrGadgetクラスのデータ構造図である。

【図２５】

メインのコントロールパネルの画面例である。

【図２６】

キャッシュ・フロー取引を作成するための初期画面例である。

【図２７】

金利スワップ取引を作成するための画面例（その１）である。

【図２８】

金利スワップ取引を作成するための画面例（その２）である。

【図２９】

金利スワップ取引を作成するための画面例（その３）である。

【図３０】

金利スワップ取引を作成するための画面例（その４）である。

【図３１】

金利スワップ取引を作成するための画面例（その５）である。

【図３２】

金利スワップ取引を作成するための画面例（その６）である。

【図３３】

金利スワップ取引を作成するための画面例（その７）である。

【図３４】

金利スワップ取引を作成するための画面例（その８）である。

【図３５】

金利スワップ取引を作成するための画面例（その９）である。

【図３６】

エクイティ・スワップ取引を作成するための画面例（その１）である。

【図３７】

エクイティ・スワップ取引を作成するための画面例（その２）である。

【図３８】

為替取引を作成するための画面例（その１）である。

【図３９】

為替取引を作成するための画面例（その２）である。

【図４０】

為替取引を作成するための画面例（その３）である。

【図４１】

"PwrGadget" 機能の初期画面例である。

【図４２】

休日除外対象都市の定義画面例である。

【図４３】

取引単位の定義画面例である。

【図４４】

イールド・カーブの定義画面例（その１）である。

【図４５】

イールド・カーブの定義画面例（その２）である。

【図４６】

プライス・カーブの定義画面例（その１）である。

【図４７】

プライス・カーブの定義画面例（その２）である。

【図４８】

仮想カーブの定義画面例である。

【図４９】

オプション取引において原資産であるスワップ取引を作成するための画面例である。

【図５０】

オプション取引の定義ウインドウの画面例である。

【図５１】

オプション取引を作成するための画面例（その１）である。

【図５２】

オプション取引を作成するための画面例（その２）である。

【図５３】

金融取引の合成手順の画面例（その１）である。

【図５４】

金融取引の合成手順の画面例（その２）である。

【図５５】

金融取引の合成手順の画面例（その３）である。

【図５６】

金融取引の合成手順の画面例（その４）である。

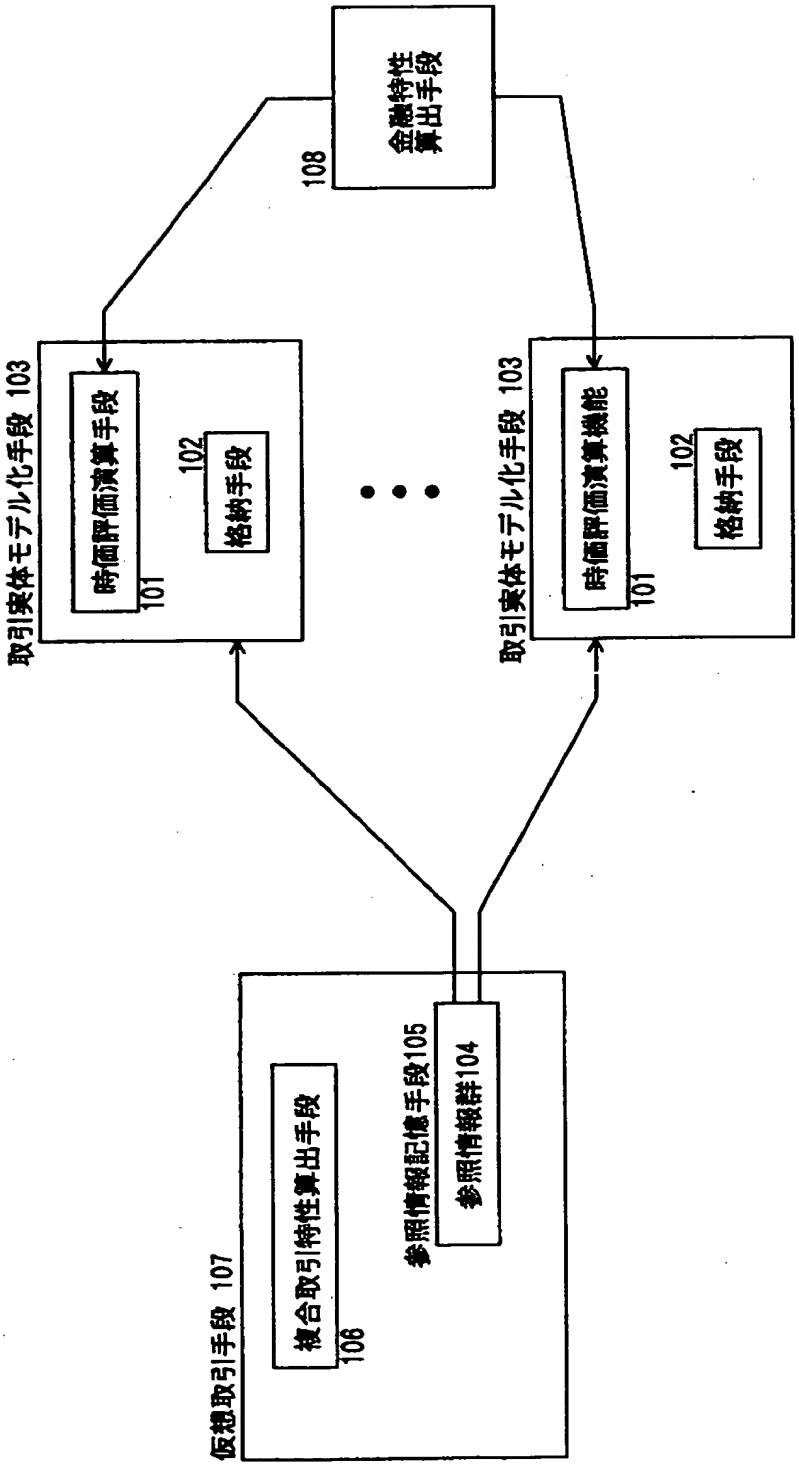
【図５７】

キャッシュ・フロー系取引を作成するための初期画面例（強調図）である。

【書類名】　　図面

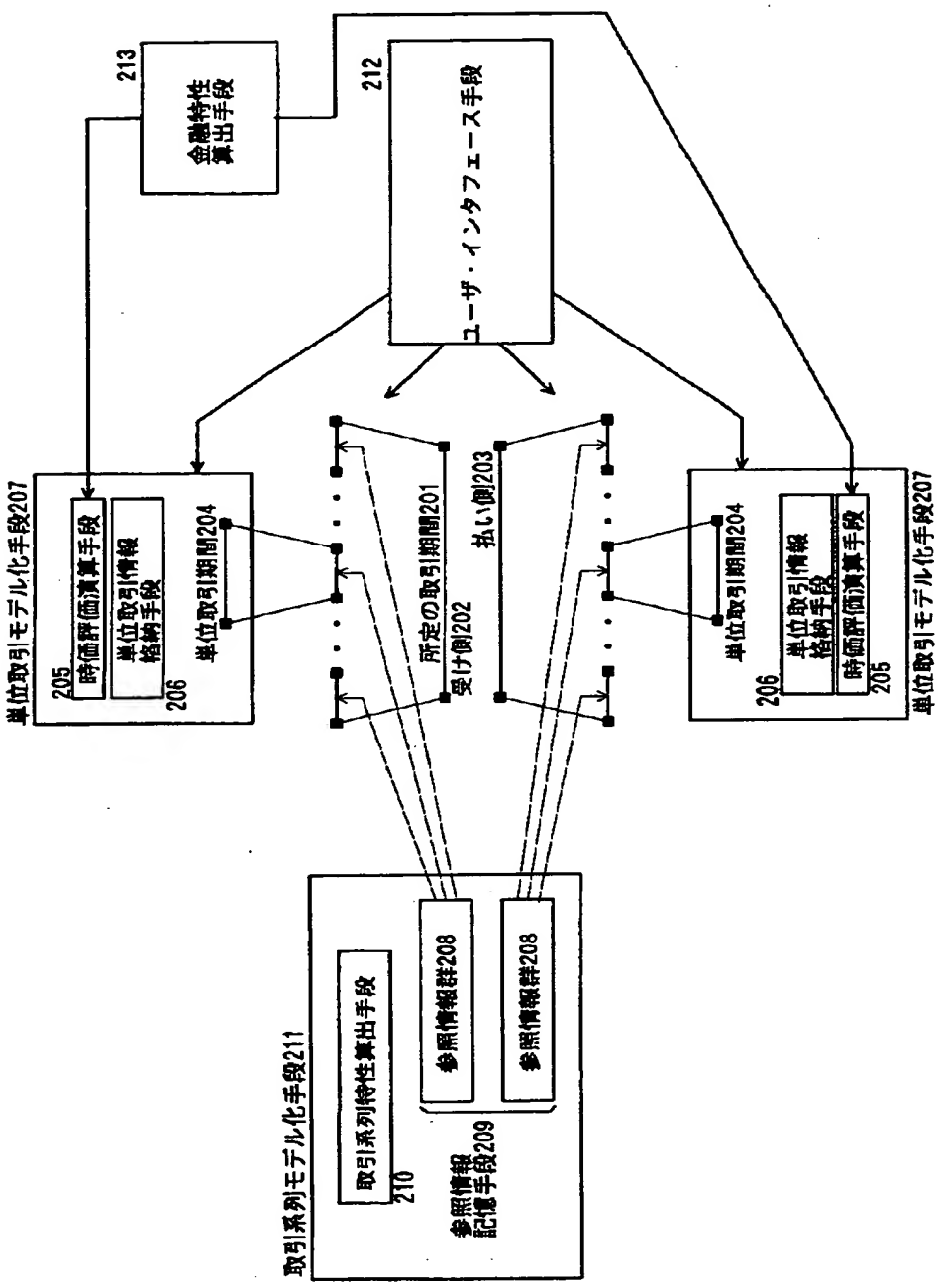
【図１】

本発明のブロック図（その１）



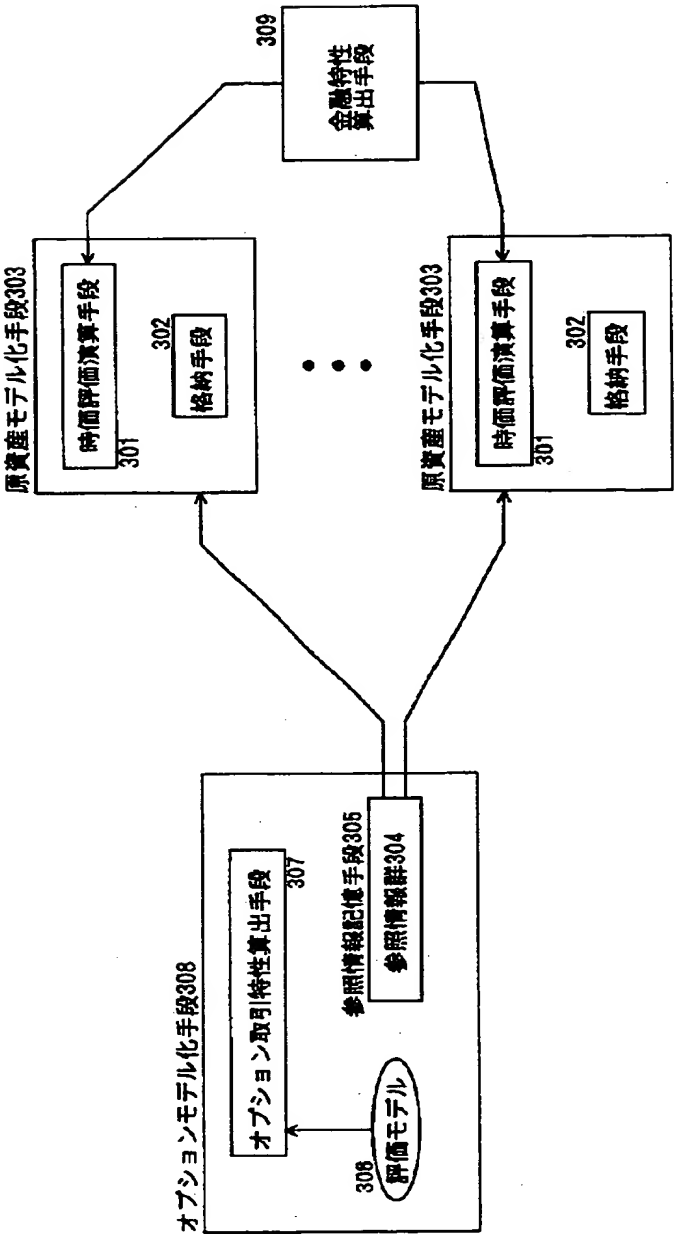
【図２】

本発明のブロック図（その２）



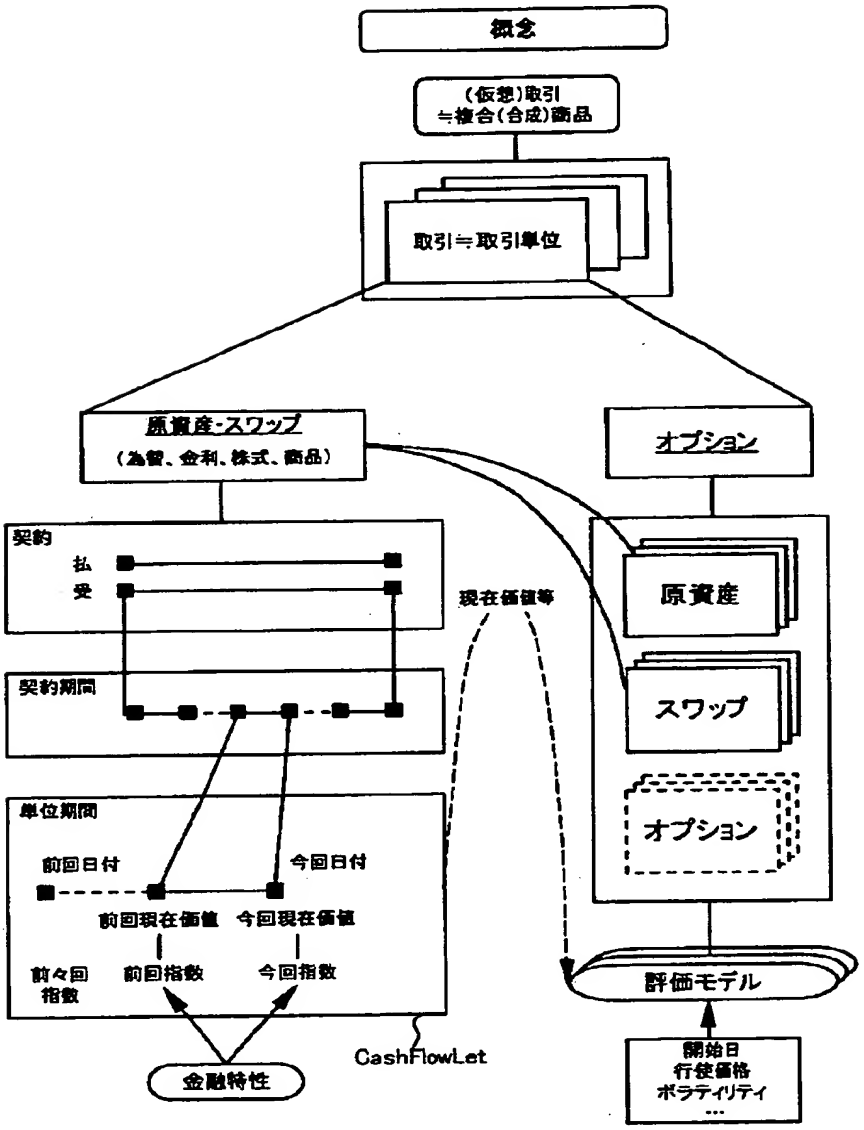
【図３】

本発明のブロック図（その３）



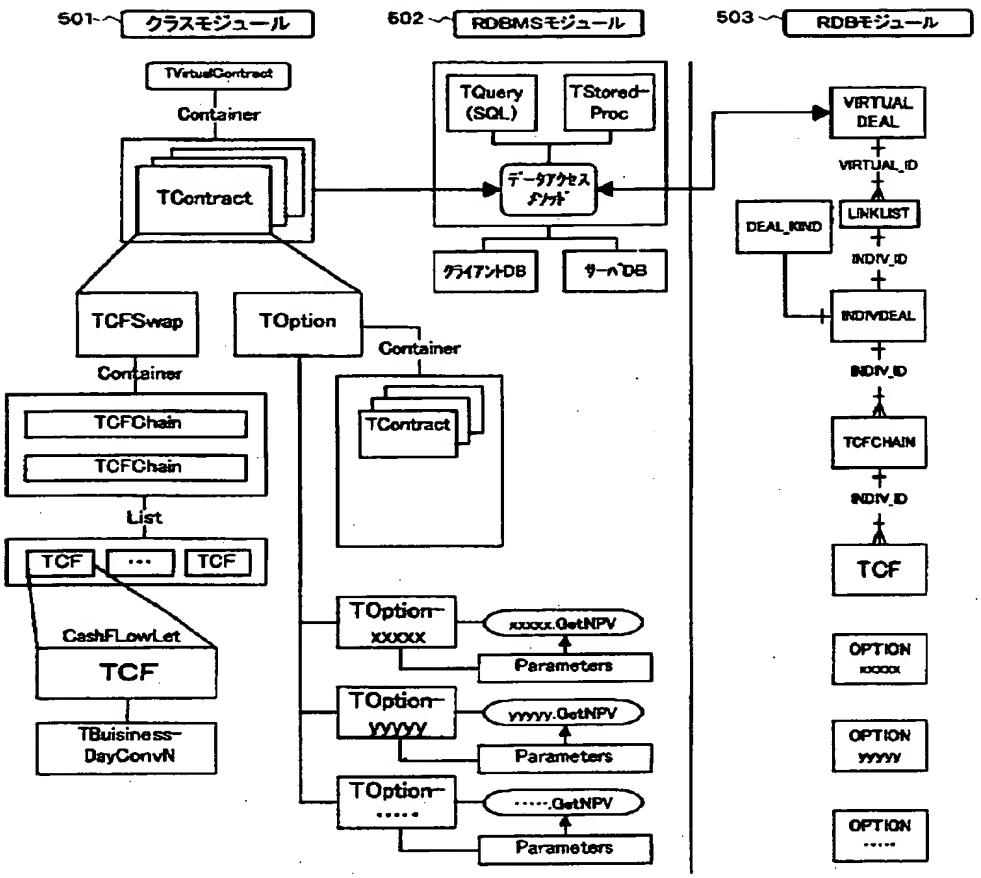
【図４】

本発明の概念図


概念
(仮想)取引
≒複合(合成)商品
取引≒取引単位
原資産-スワップ
(為替、金利、株式、商品)
オプション
契約
払
受
契約期間
単位期間
前回日付
今回日付
前回現在価値
今回現在価値
前々回
指数
前回指数
今回指数
金融特性
CashFlowLet
現在価値等
原資産
スワップ
オプション
評価モデル
開始日
行使価格
ボラティリティ
...

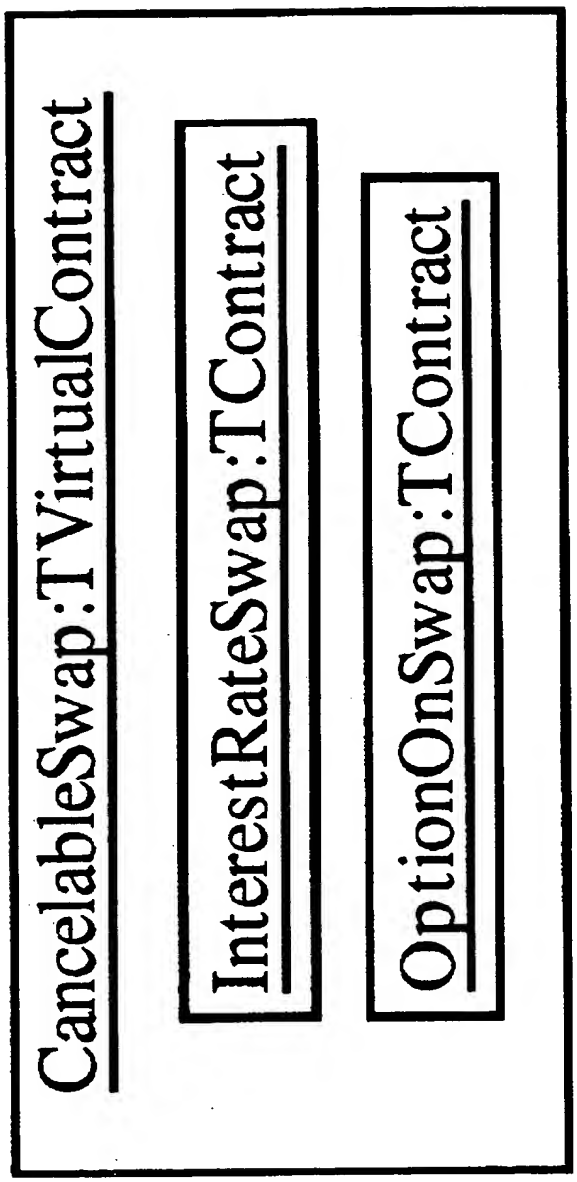
【図５】

本発明の実施の形態の全体構成図



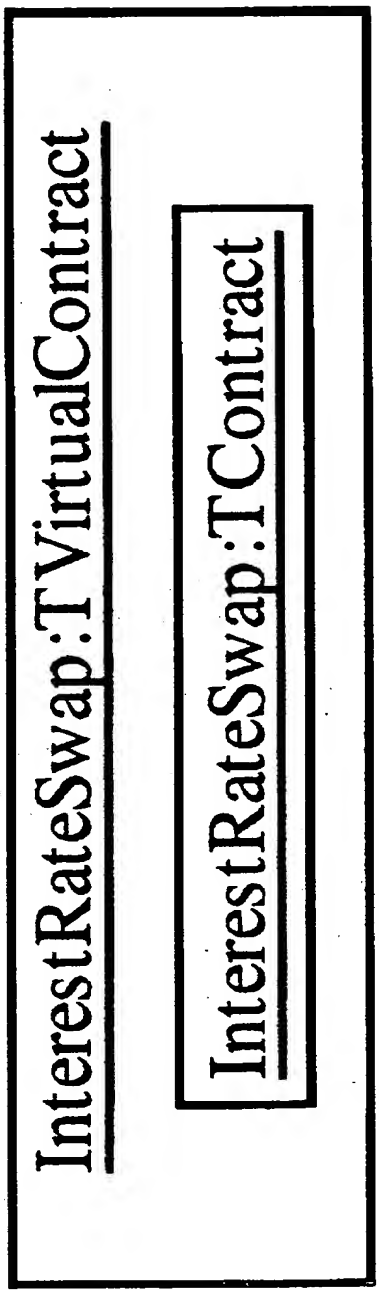
【図６】

解約権付きスワップの
仮想取引管理の説明図



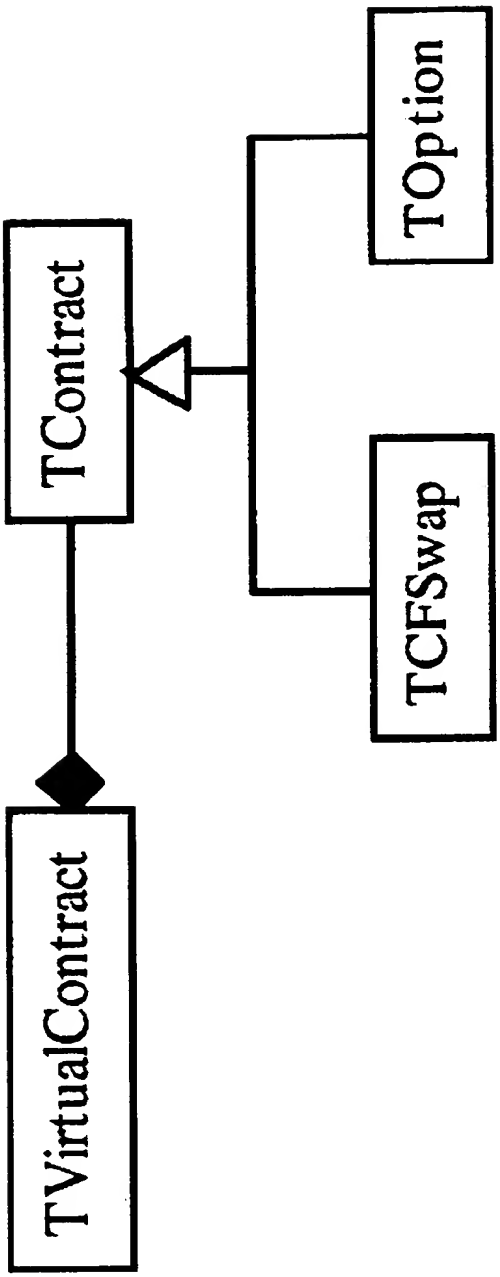
【図７】

プレーンバニラ・スワップの
仮想取引管理の説明図



【図８】

仮想取引を実現するクラス構造図



【図９】

データ圧縮の説明図（その１）

InterestRateSwap 001

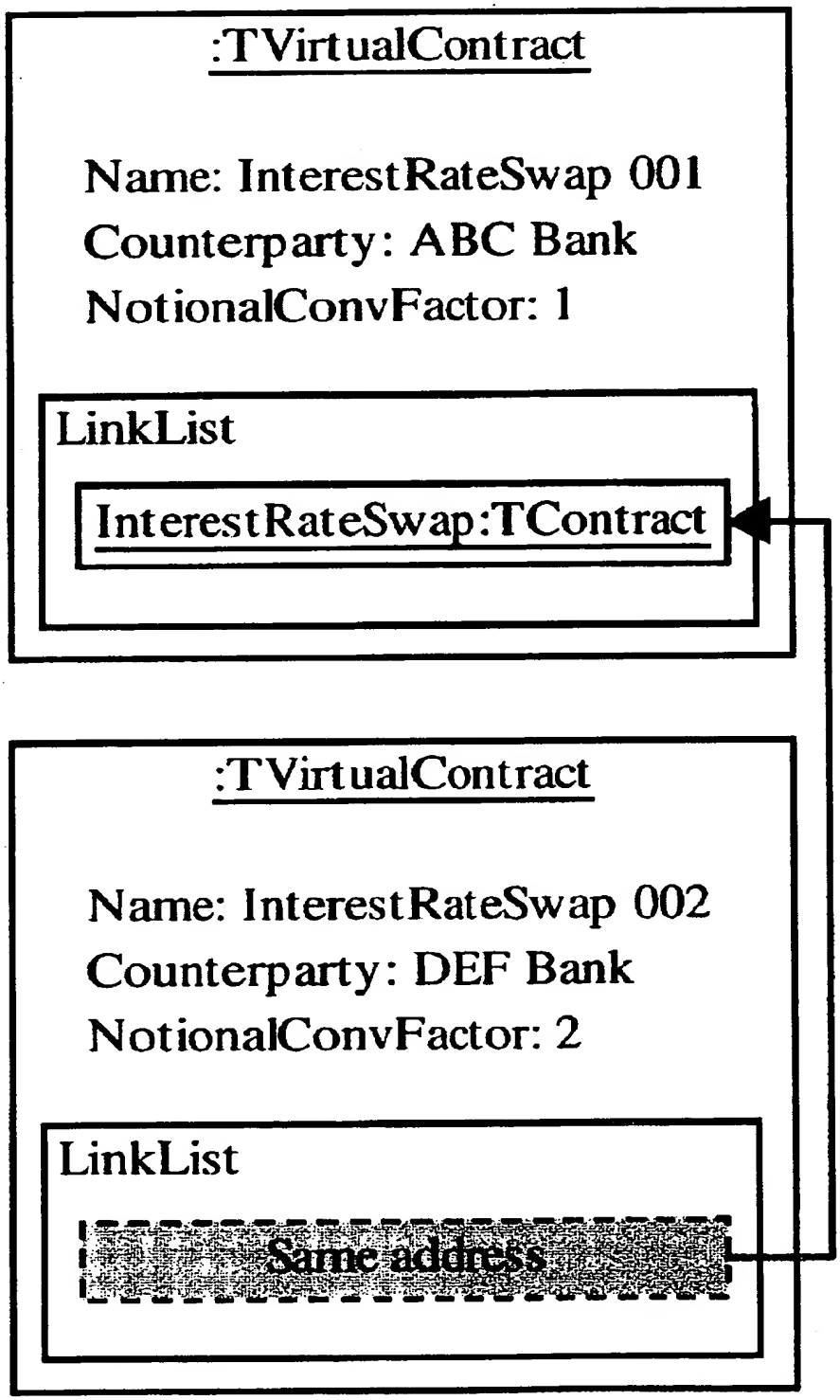
取引種：金利スワップ
契約先：ABC 銀行
実効日：19xx 年 xx 月 xx 日
契約期間：実効日より y 年間
約定レート：z%
想定元本：10 億円

InterestRateSwap 002

取引種：金利スワップ
契約先：DEF 銀行
実効日：19xx 年 xx 月 xx 日
契約期間：実効日より y 年間
約定レート：z%
想定元本：20 億円

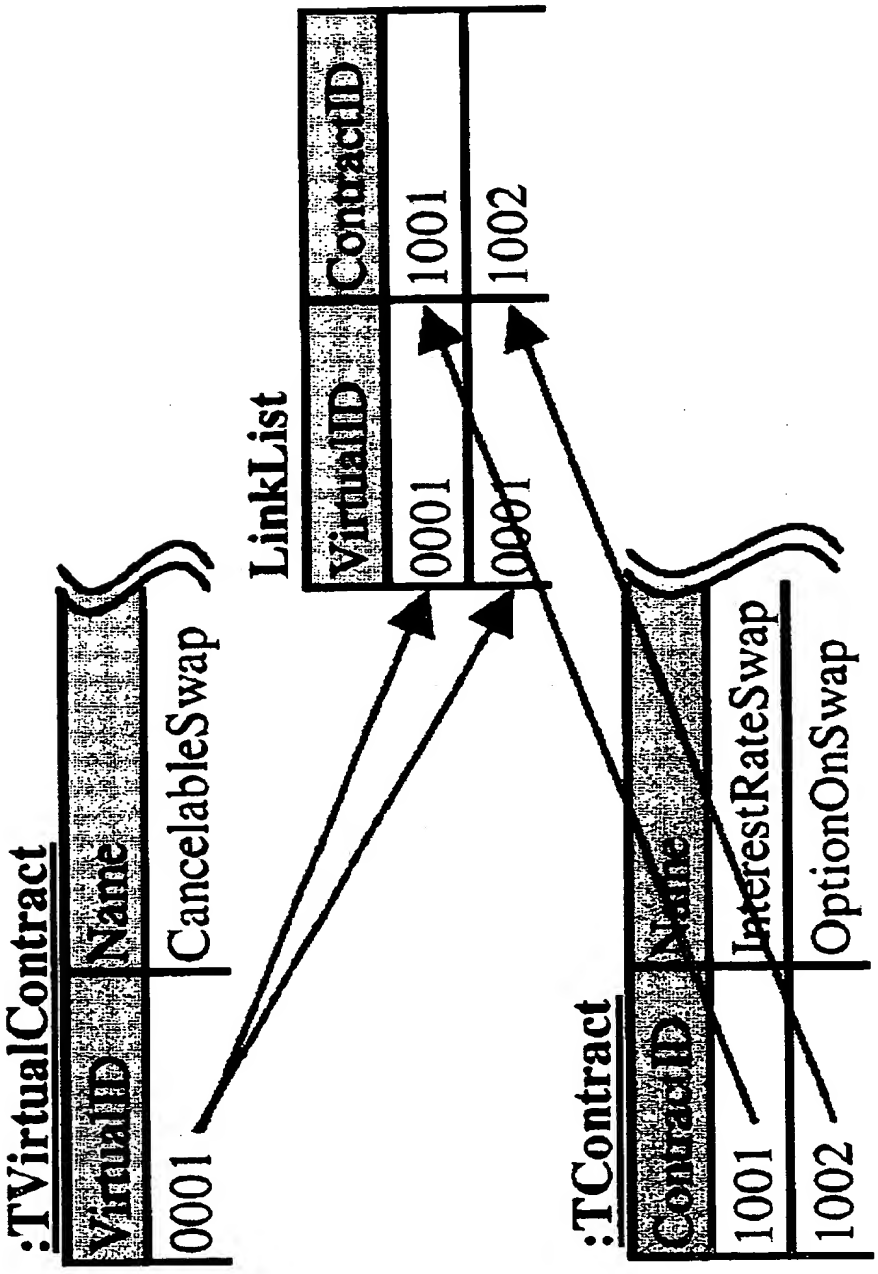
【図１０】

データ圧縮の説明図（その２）



【図１１】

ＲＤＢＭＳ上での仮想取引の実現構造図



【図１２】

RDB上のテーブル構成図(その1)

VIRTUALDEAL		
フィールド名	型	備考
VIRTUAL_ID	INTEGER	仮想取引インスタンス識別コード(自動生成)
FREE_ID1	VARCHAR	利用者定義ID 取引種類コードとして利用することを推奨
FREE_ID2	VARCHAR	利用者定義ID 取引分類コードとして利用することを推奨
FLDR1_NAME	VARCHAR	取引格納親フォルダの名称
FLDR2_NAME	VARCHAR	取引格納子フォルダの名称
FREE_NAME	VARCHAR	利用者定義名称 個別取引に対する名称として利用することを推奨
REGISTDATE	DATE	登録日付・時刻(自動生成)
CTRPTY_ID	VARCHAR	取引先ID

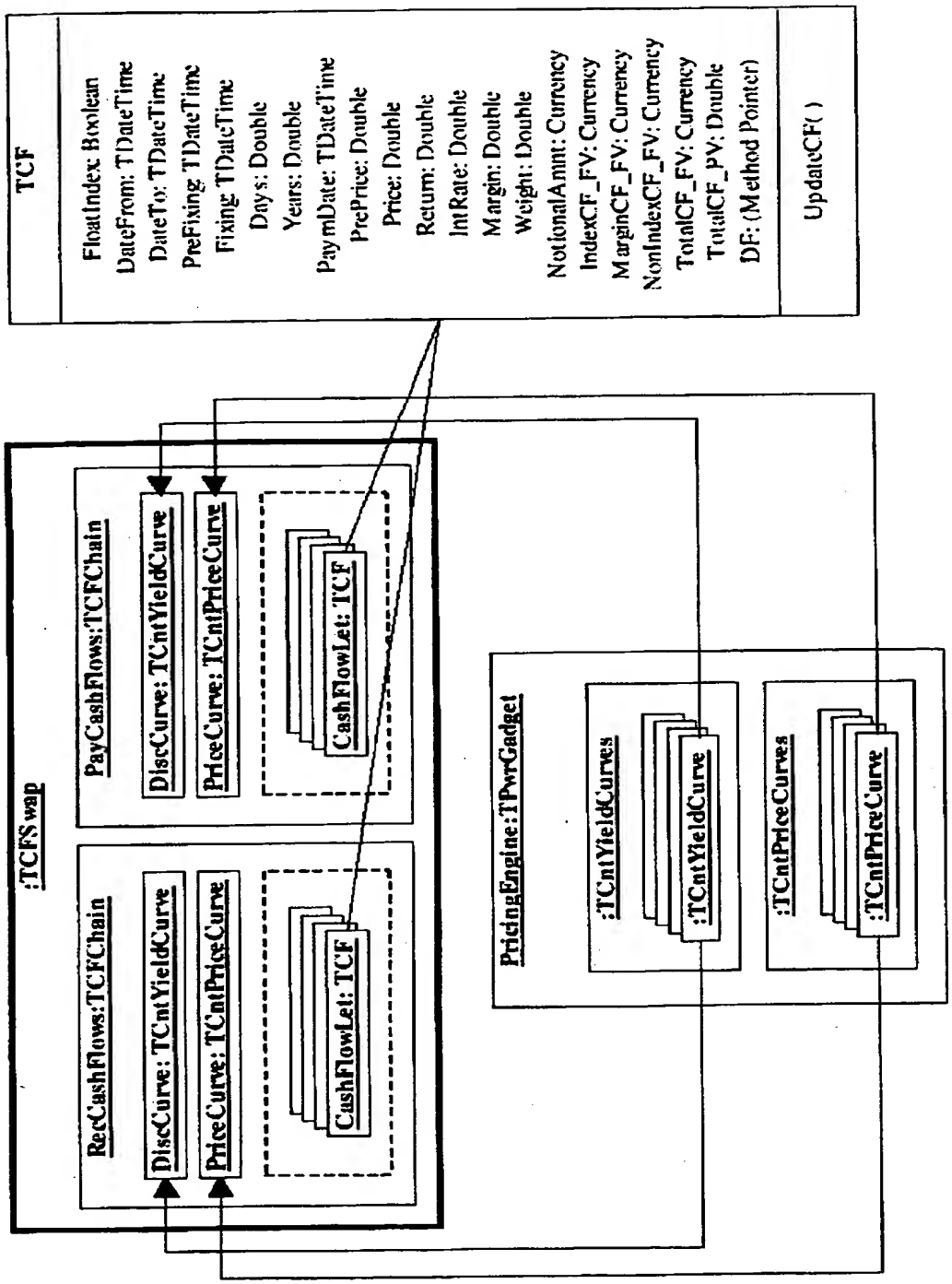
INDIVDEAL		
フィールド名	型	備考
INDIV_ID	INTEGER	実体取引インスタンス識別コード(自動生成)
KIND	VARCHAR	取引種名

DEAL_KIND		
フィールド名	型	備考
KIND	VARCHAR	取引種名
TYPE	CHAR	クラス・ライブラリ内のクラス識別コード

LINKLIST		
フィールド名	型	備考
VIRTUAL_ID	INTEGER	仮想取引インスタンス識別コード(自動生成)
INDIV_ID	INTEGER	実体取引インスタンス識別コード(自動生成)

【図１３】

TCFSwapインスタンス・イメージとTCFクラスのデータ構造図



【図１４】

TCFChainクラスの属性概要を示す図表

属性	説明
PaymCenters	受払日の休日除外対象都市リスト
FixCenters	値決め日の休日除外対象都市リスト
EDate	実効日
TDate	満期日
FDate	次回受払日
BDConv	休日除外方式 (Preceding / Following / Modified など)
EndEnd	月末ロール指定
AdjTDate	満期日休日除外指定
CFFrq	受払頻度
ResetFrq	リセット頻度
FixingOffset	値決め日オフセット日付
FixingBasis	値決め日決定基準
PayIn	受払日決定基準
CFList	TCF インスタンス・リストへの参照
AppliedUnit	適用通貨・証券・商品等の単位
DiscCurve	割引カーブ
PriceCurve	価格カーブ
IniPrincAmnt	初期元本交換額
FinPrincAmnt	終期元本交換額
VarNotional	Variable Notional Principal 指定
AddMargin	マージン繰り込み方式指定
NotionalAmnt	想定元本額
IndexType	指数種（IntRate/ Return/ Price）
FloatIndex	変動／固定指数区分
IndexVal	指数値
IndexUnit	指数単位（% / b.p. / decimal）
DayCount	日割り計算方式 (Bond / Act/360/ Act/Act, Act/365Fixed, EuroBond など)
Rounding	四捨五入／切り捨て指定
DecPlaces	四捨五入／切り捨て桁指定

【図１５】

RDB上のテーブル構成図(その2)

TCFCHAIN		
フィールド名	*型*	*備考*
TCFCHAIN_ID	INTEGER	TCFCHAINインスタンス識別コード
INDIV_ID	INTEGER	実体取引インスタンス識別コード
SIDE_ID	CHAR	"受け／払い"区分コード
EDATE	DATE	EffectiveDate
TDATE	DATE	TerminationDate
FDATE	DATE	Stub 指定時の FirstDate
BDCONV	SMALLINT	BusinessDayConvention識別コード
ENDEND	CHAR	EndRoll 指定(論理型)
ADJTDATE	CHAR	TerminationDate 修正指定(論理型)
CFFRQ	SMALLINT	PaymentFrequency
RESETFRQ	SMALLINT	ResetFrequency
FIXINGOFFSET	INTEGER	FixingDate オフセット日数
FIXINGBASIS	SMALLINT	FixingDate オフセット起点
PAYIN	SMALLINT	Payment 方式 (Advance/Arrear)
UNIT_CODE	VARCHAR	受払い通過
DISCCURVE	DATE	割引カーブ・インスタンス識別コード 注) 日付値をコードとして利用
PRICECURVE	DATE	割引カーブ・インスタンス識別コード 注) 日付値をコードとして利用
INIPRINCAMNT	NUMERIC(15,2)	実交換元本 (Initial)
FINPRINCAMNT	NUMERIC(15,2)	実交換元本 (Final)
VARNOTIONAL	CHAR	VariableNotionalPrincipal 指定(論理型)
ADDMARGIN	CHAR	マージン繰り込み指定(論理型)
NOTIONALAMNT	NUMERIC(15,2)	NotionalAmount
INDEXTYPE	SMALLINT	指数種規定 (Interest/Return/Price)
FLOATINDEX	CHAR	変動指数指定(論理型)
INDEXVAL	DOUBLE	指数値 (InterestRate,ReturnValue,Price)
INDEXUNIT	SMALLINT	指数単位 (%/b.p./dec)
DAYCOUNT	SMALLINT	DayCountBasis (Act_365Fix/Act_Act/Act_360/30E_360/30_360)
ROUNDING	CHAR	四捨五入指定(論理型)
DECPLACES	SMALLINT	丸め位置(小数点下桁数)

【図１６】

TCFクラスの属性概要を示す図表

属性	説明
FloatIndex	変動／固定指数区分
DateFrom	計算期間開始日
DateTo	計算期間終了日
PreFixing	前期値決め日
Fixing	当期値決め日
Days	計算期間日数
Years	計算期間年数
PaymDate	受払日
PrePrice	前期価格指数
Price	当期価格指数
Return	収益率
IntRate	金利値
Margin	マージン値
Weight	加重指数
NotionalAmnt	想定元本金額
IndexCF_FV	指数に基づく受払金額
MarginCF_FV	マージンに基づく受払金額
NonIndexCF_FV	指数とは独立した受払金額
TotalCF_FV	受払総額
TotalCF_PV	受払総額現在価値
DF	ディスカウント・ファクター（GetFactor へのメソッド・ポインタ）

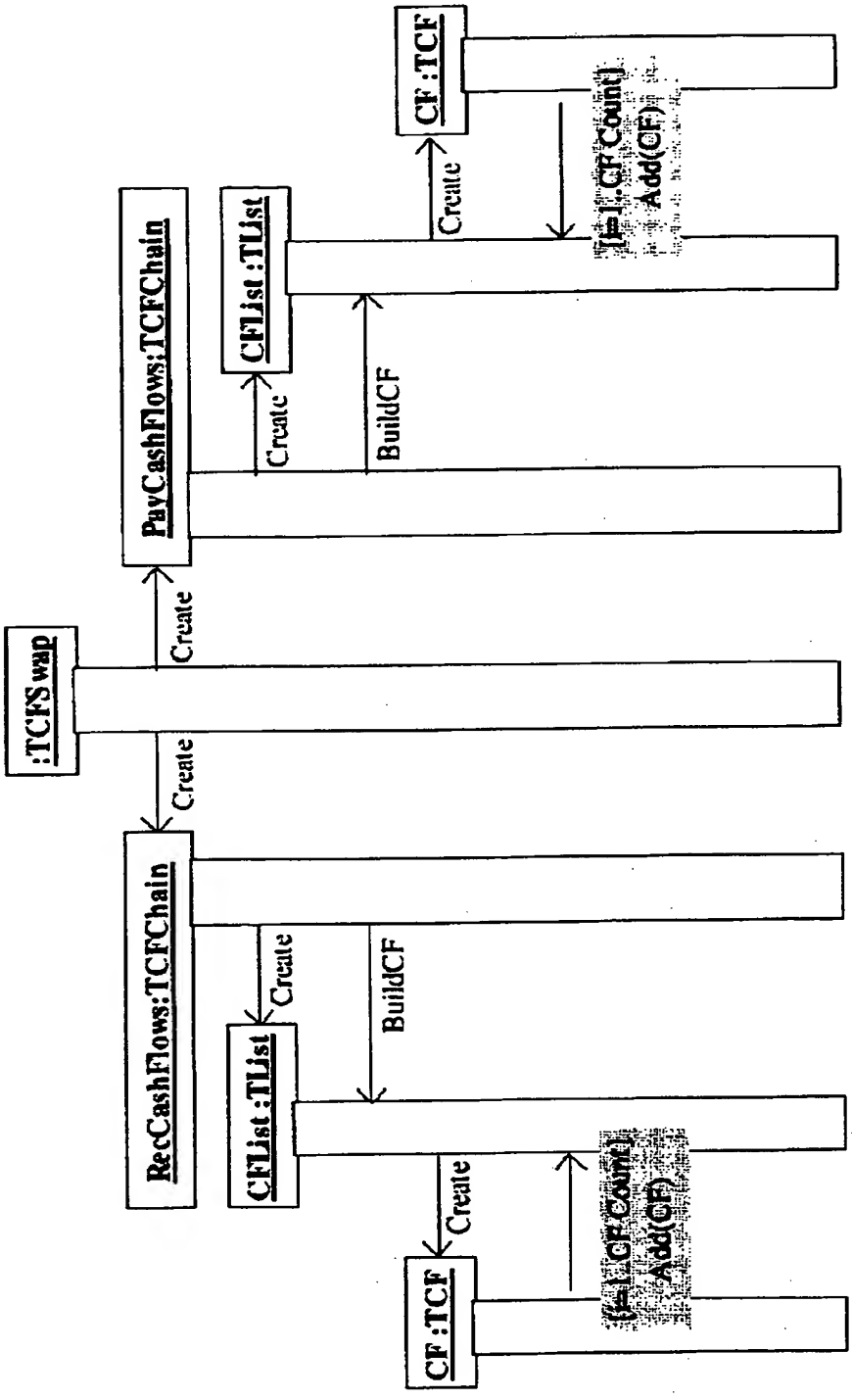
【図１７】

RDB上のテーブル　成図(その3)

TCF		
フィールド名	*型*	*備考*
TCF_ID	INTEGER	TCFインスタンス識別コード
INDIV_ID	INTEGER	実体取引インスタンス識別コード
SIDE_ID	CHAR	"受け／払い"区分コード
FLOATINDEX	CHAR	変動指数指定(論理型)
DATEFROM	DATE	CalculationPeriod 開始日
DATETO	DATE	CalculationPeriod 終了日
DAYS	INTEGER	期間日数
YEARS	DOUBLE	YearFraction
PAYMDATE	DATE	受払い日
PREFIXING	DATE	前回 FixingDate 注) FloatIndex の場合のみ有効
FIXING	DATE	FixingDate 注) FloatIndex の場合のみ有効
PREPRICE	DOUBLE	前回価格
PRICE	DOUBLE	価格
RETURN	DOUBLE	収益率
INTRATE	DOUBLE	金利値 注) FixedRate の場合のみ保持
MARGIN	DOUBLE	マージン値
WEIGHT	DOUBLE	掛け目
NOTIONALAMNT	NUMERIC(15,2)	NotionalAmount 注) FixedNotional の場合のみ保持
INDEXCF_FV	NUMERIC(15,2)	指数関連キャッシュ・フロー(将来価値) 注) FixedIndex の場合のみ有効
MARGINCF_FV	NUMERIC(15,2)	マージン・キャッシュ・フロー(将来価値)
NONINDEXCF_FV	NUMERIC(15,2)	指数関連外キャッシュ・フロー(将来価値)
TOTALCF_FV	NUMERIC(15,2)	当該受払い日における受払い総額(将来価値) 注) FixedIndex の場合のみ有効

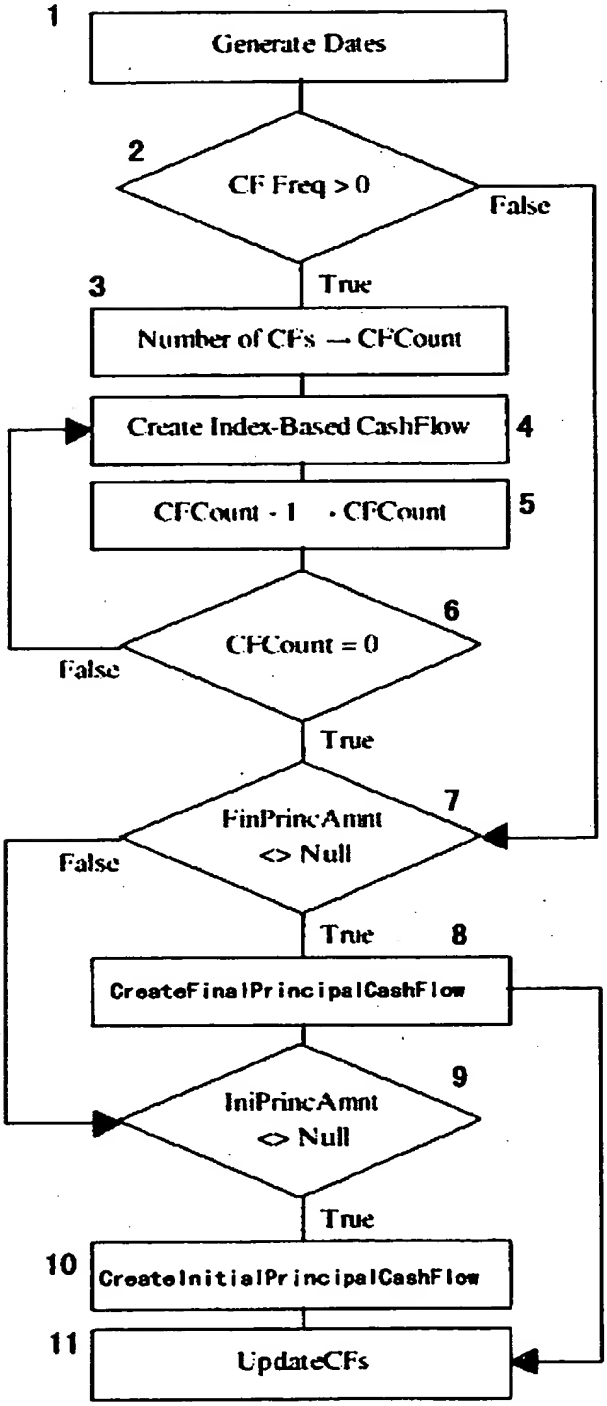
【図１８】

TCFSwapインスタンスの生成の手順の説明図



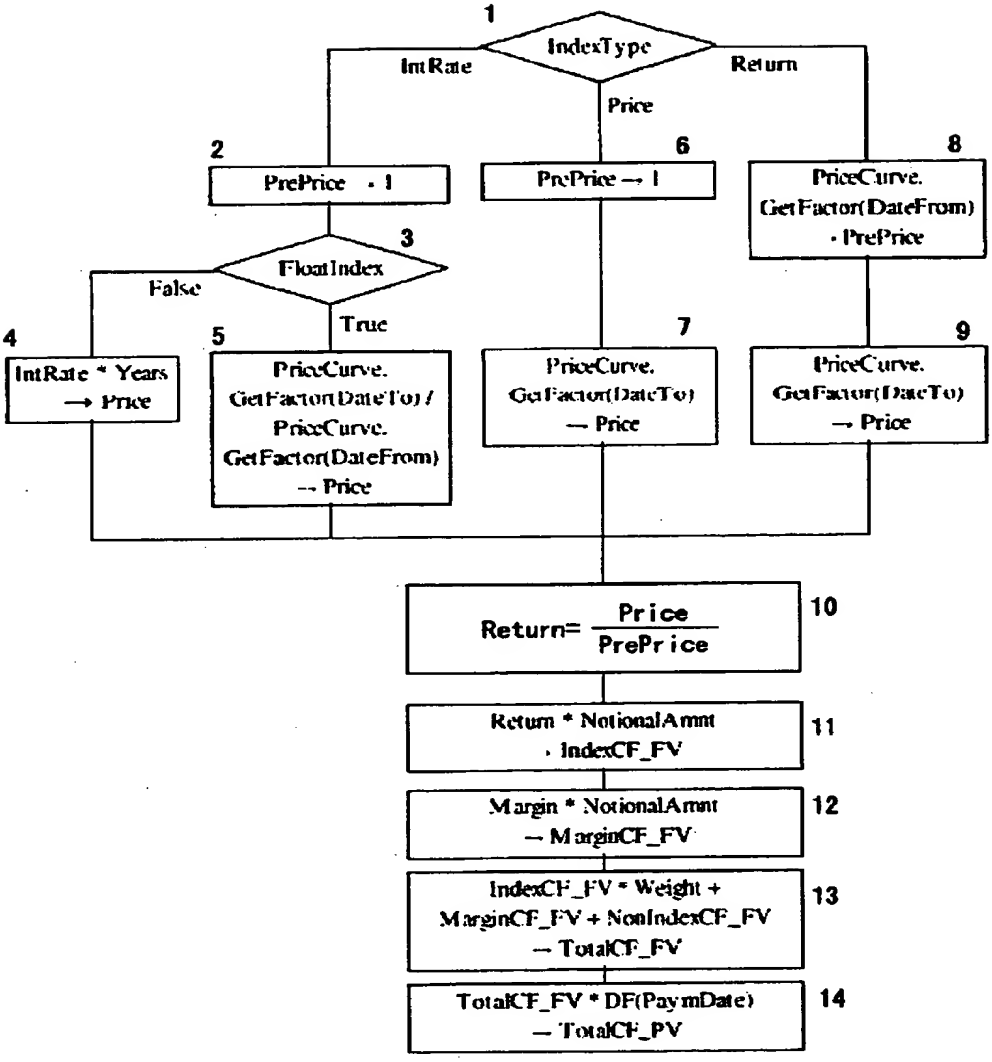
【図１９】

TCFChain.BuildCFメソッドの処理フロー



【図２０】

TCF.Updateメソッドの処理フロー



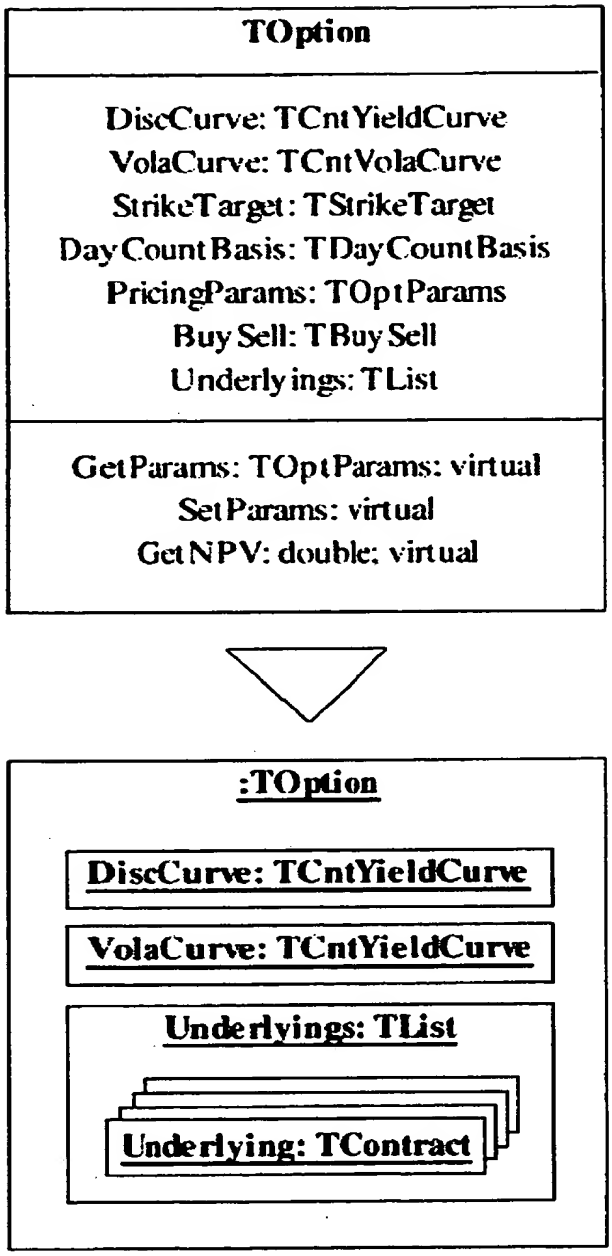
【図２１】

取引の分類を示す図表

分類番号	取引事例
1	為替、株式、商品のスポット／アウトライト・フォワード
2	割引債、株式・商品現先
3	金利スワップ、クーポン・スワップ
4	利付債、現金貸借、株式・商品現先
その他	通貨スワップ

【図２２】

TOptionクラスのデータ構造とインスタンス・イメージを示す図



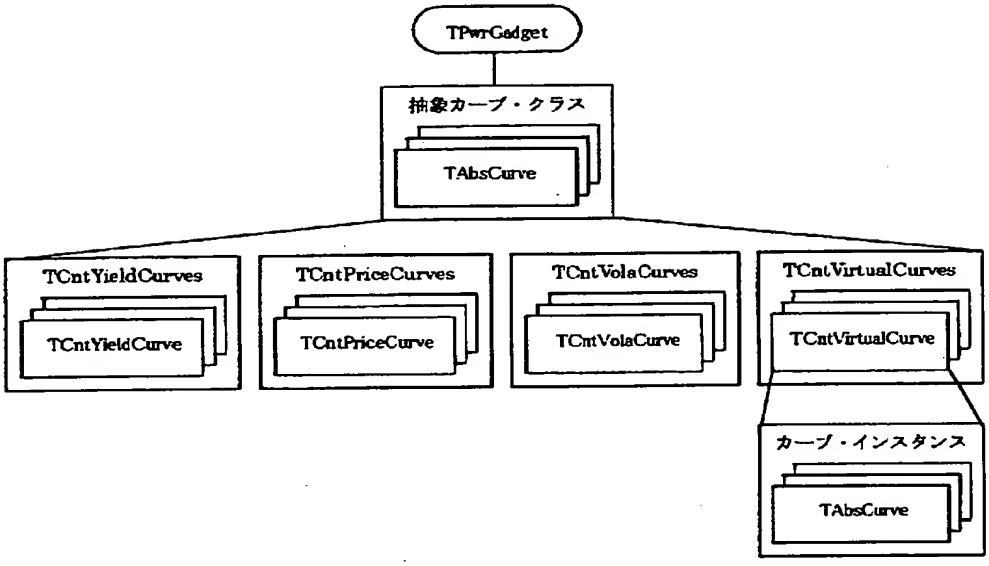
【図２３】

TOptionの属性とメソッドの概要を示す図表

属性	説明
DiscCurve	ディスカウント・カーブ
VolaCurve	ボラティリティ・カーブ
StrikeTarget	ストライク・ターゲット（オプションの権利行使価格と対比させる原資産の可変パラメータを指定する属性）
DayCountBasis	日割り計算方式
PricingParams	価格評価メソッドの実行に必要なパラメータ群
BuySell	買／売 区分
Underlyings	原資産を格納するコンテナ（リンク・リスト）
メソッド	説明
GetParams	属性PricingParamsからパラメータ群を取得
SetParams	属性PricingParamsにパラメータ群を設定
GetNPV	価格評価（時価評価）を実行

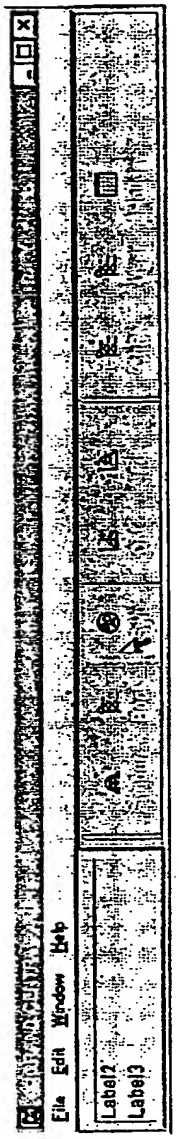
【図２４】

TpwrGadget クラスのデータ構造図



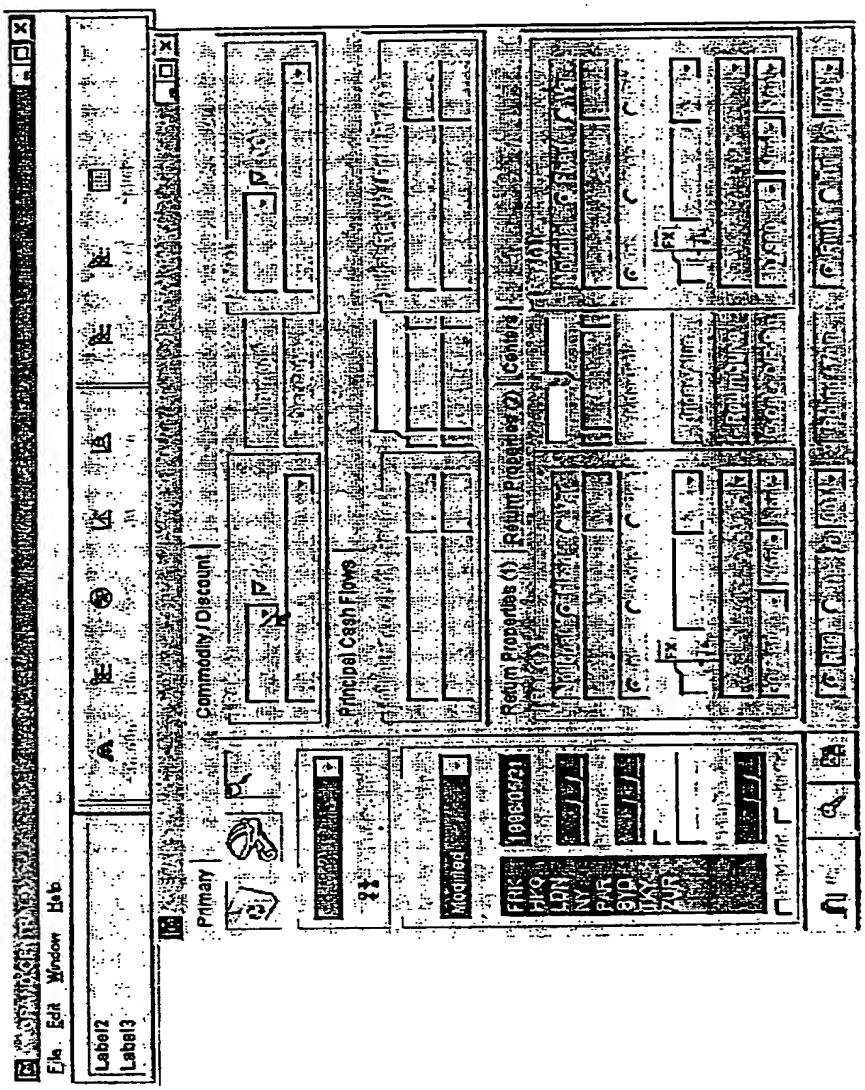
【図２５】

メインのコントロールパネルの画面例



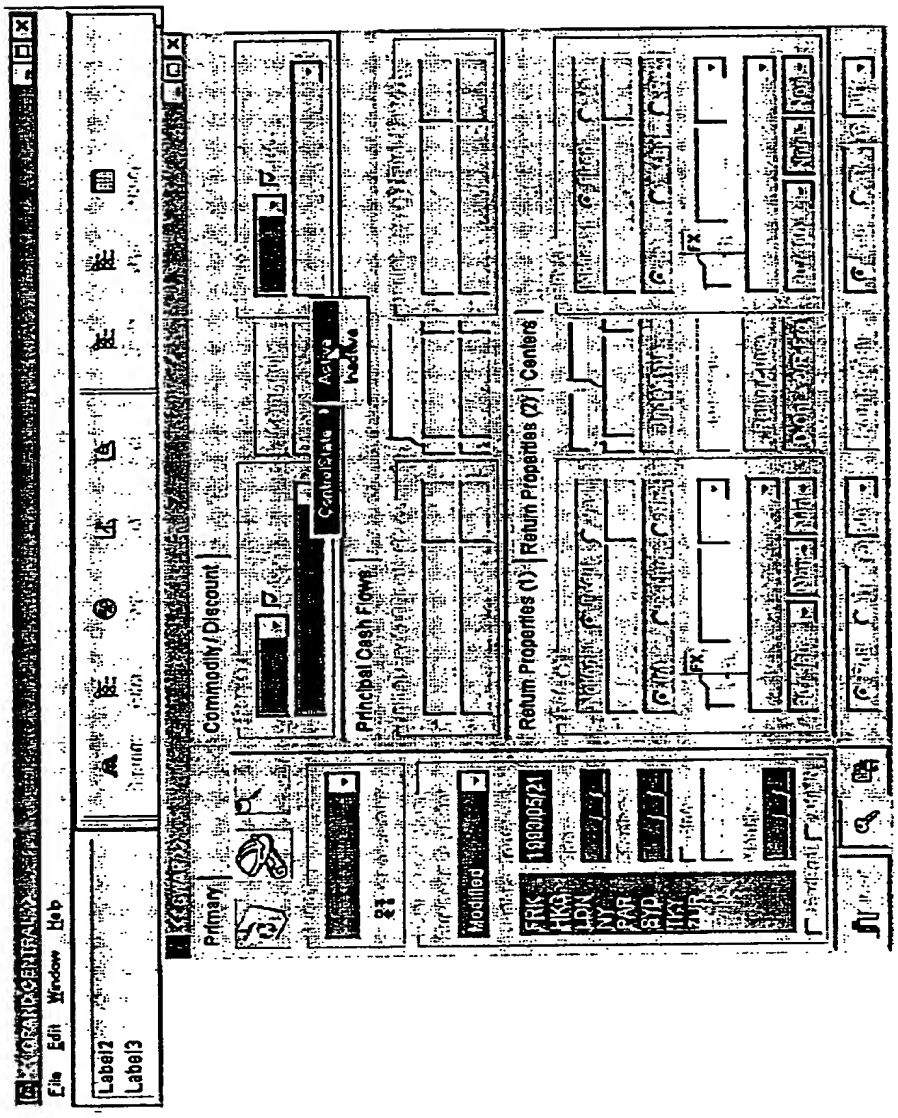
【図２６】

キャッシュ・フロー取引を作成するための初期画面例



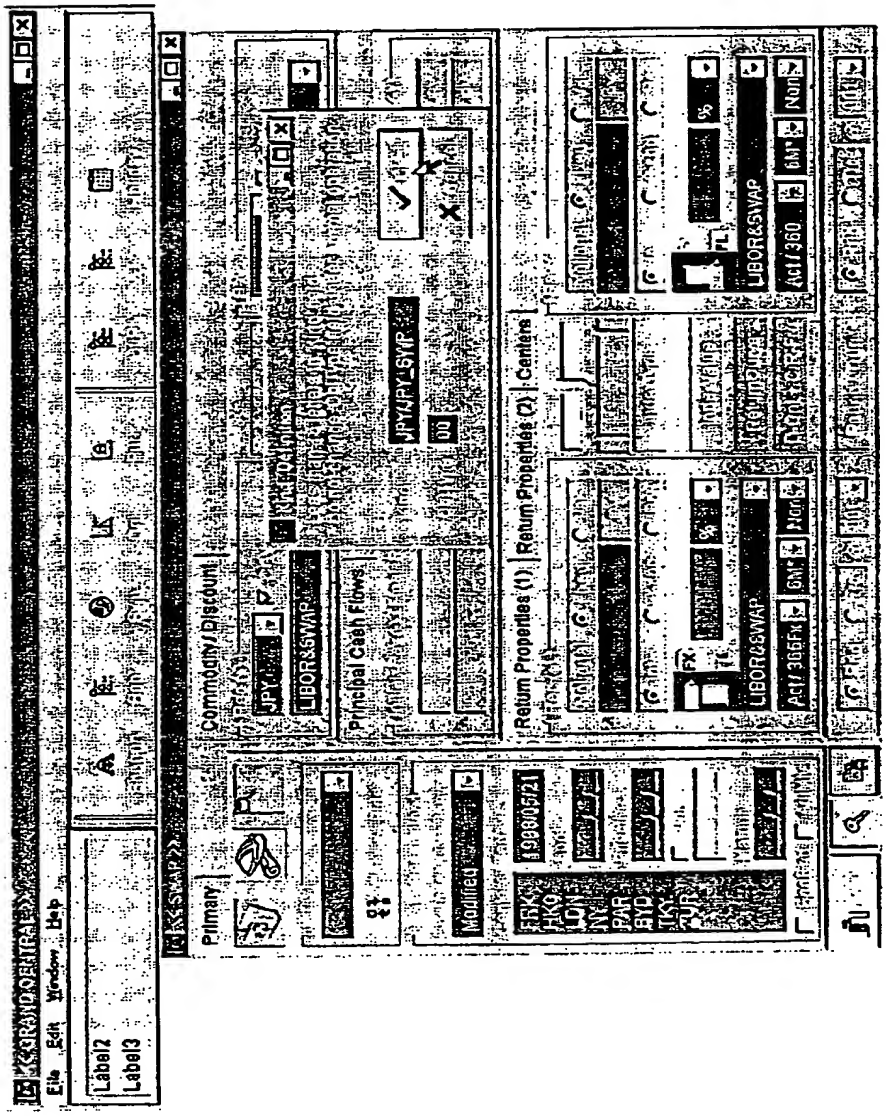
【図２７】

金利スワップ取引を作成するための画面例（その１）



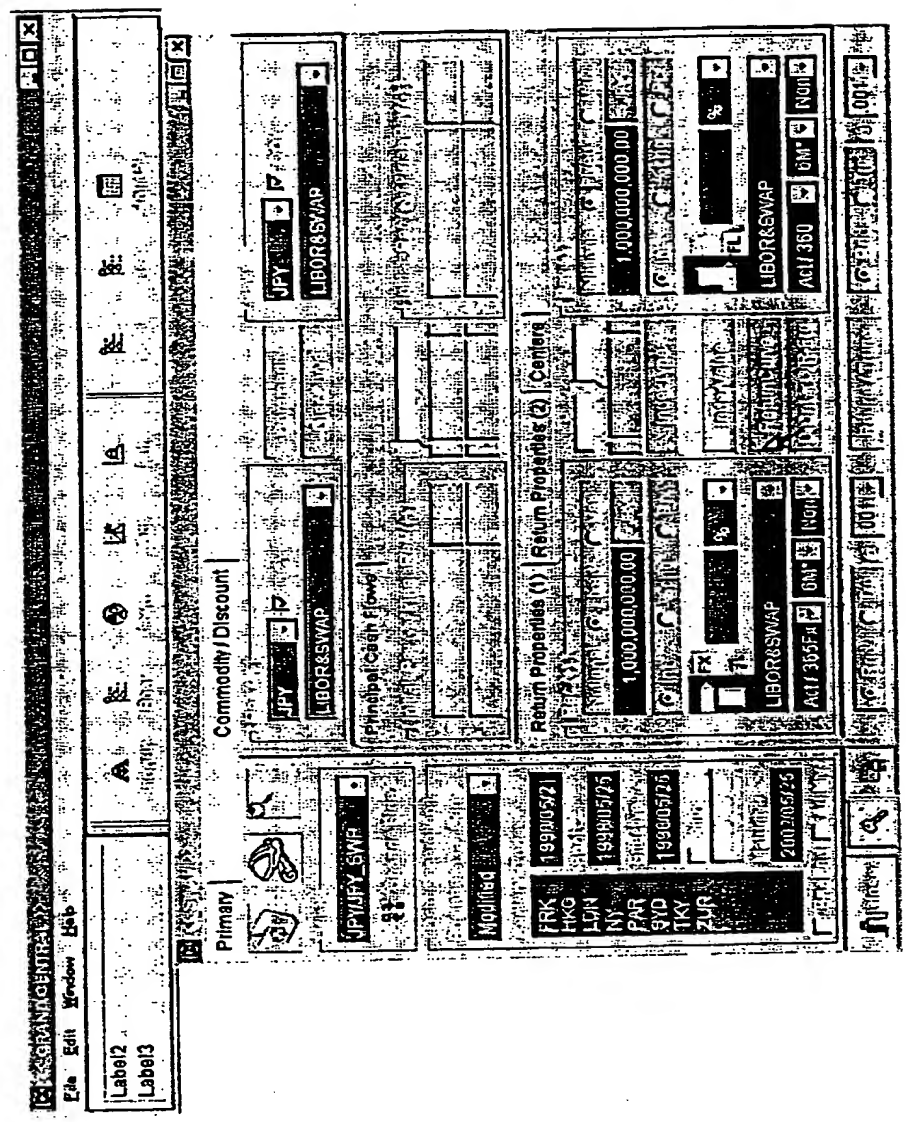
【図２８】

金利スワップ取引を作成するための画面例（その２）



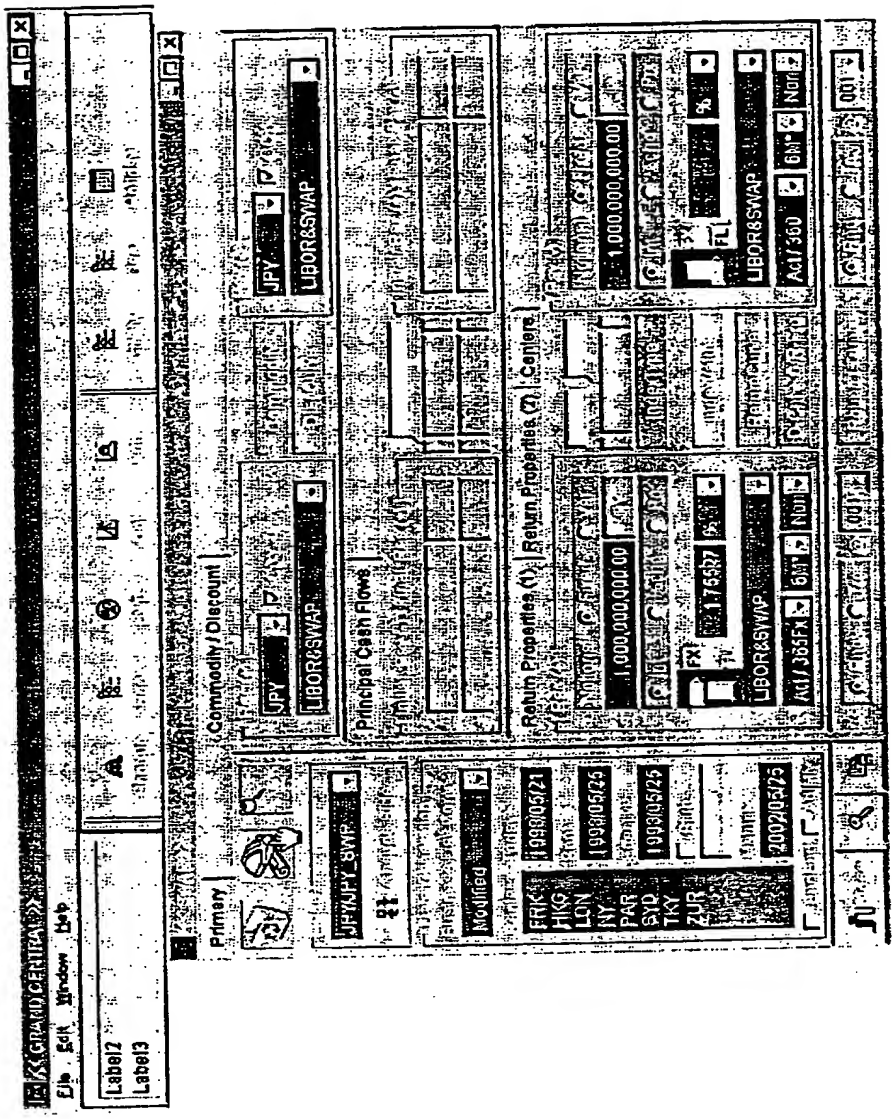
【図２９】

金利スワップ取引を作成するための画面例（その３）



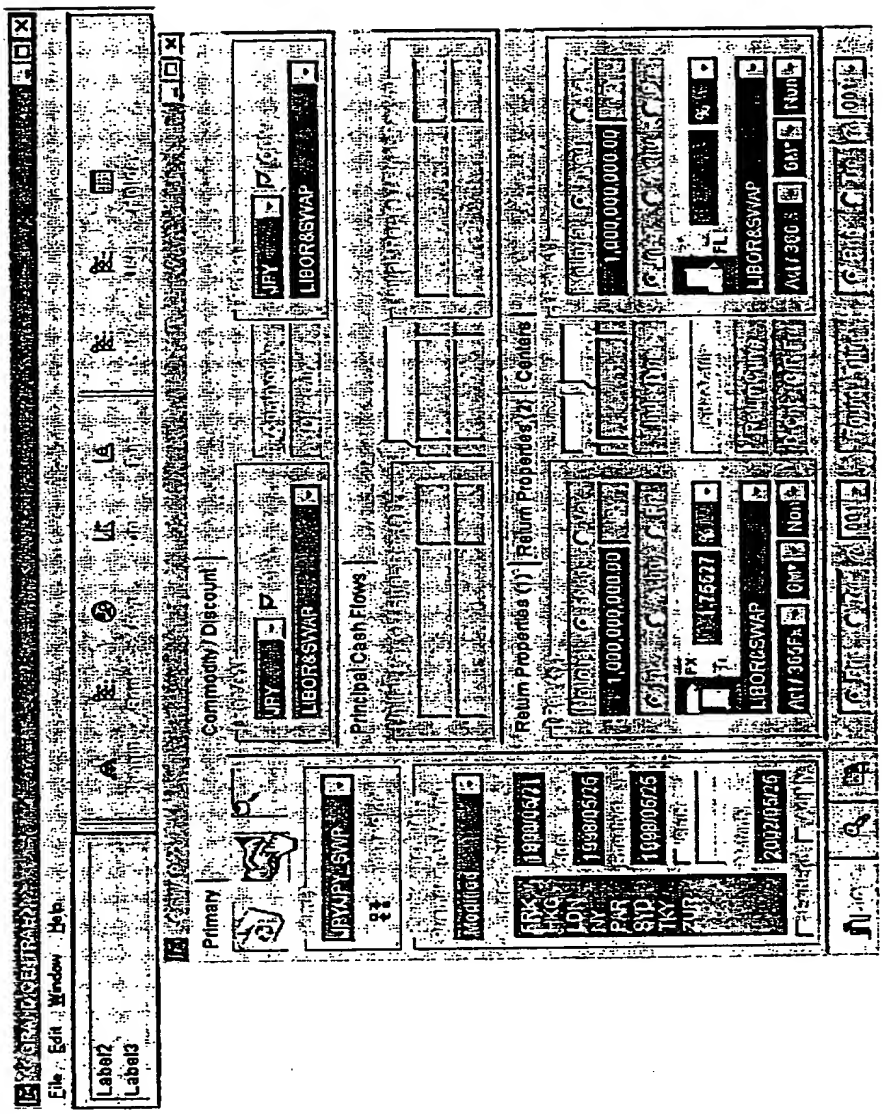
【図３０】

金利スワップ取引を作成するための画面例（その４）



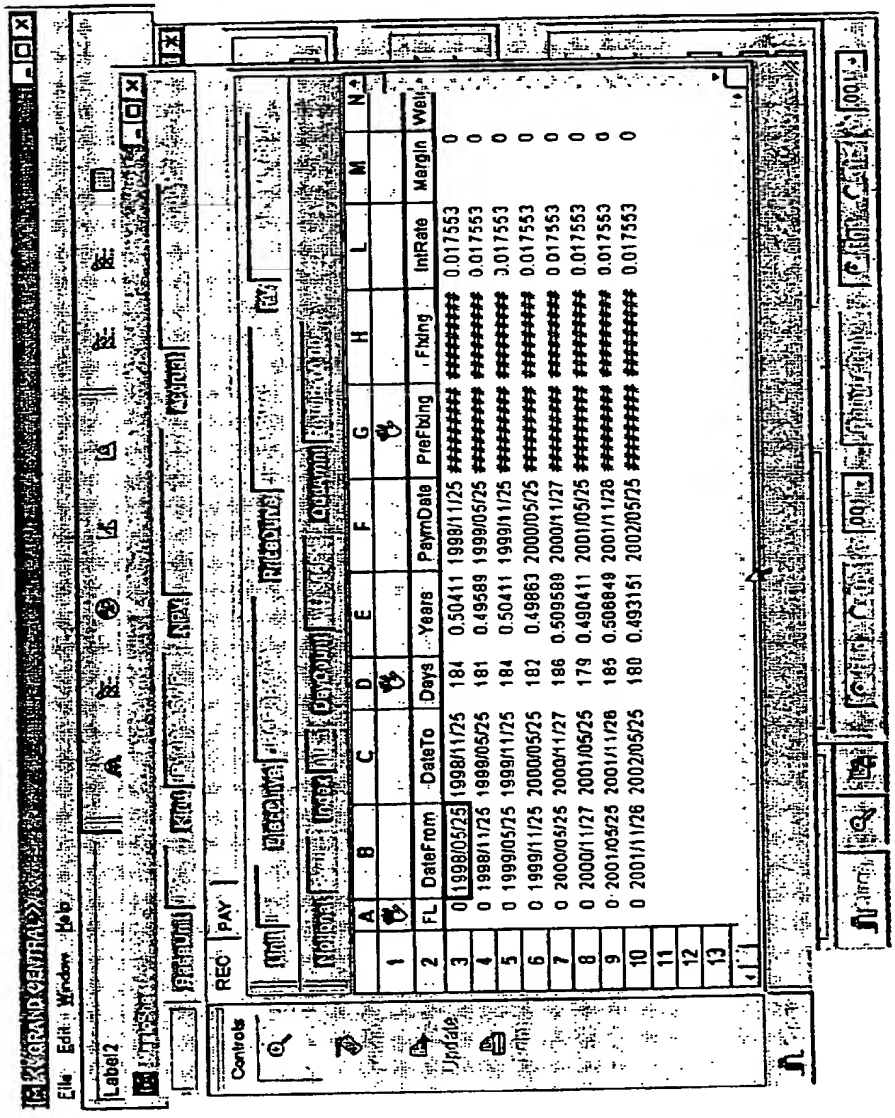
【図３１】

金利スワップ取引を作成するための画面例（その５）



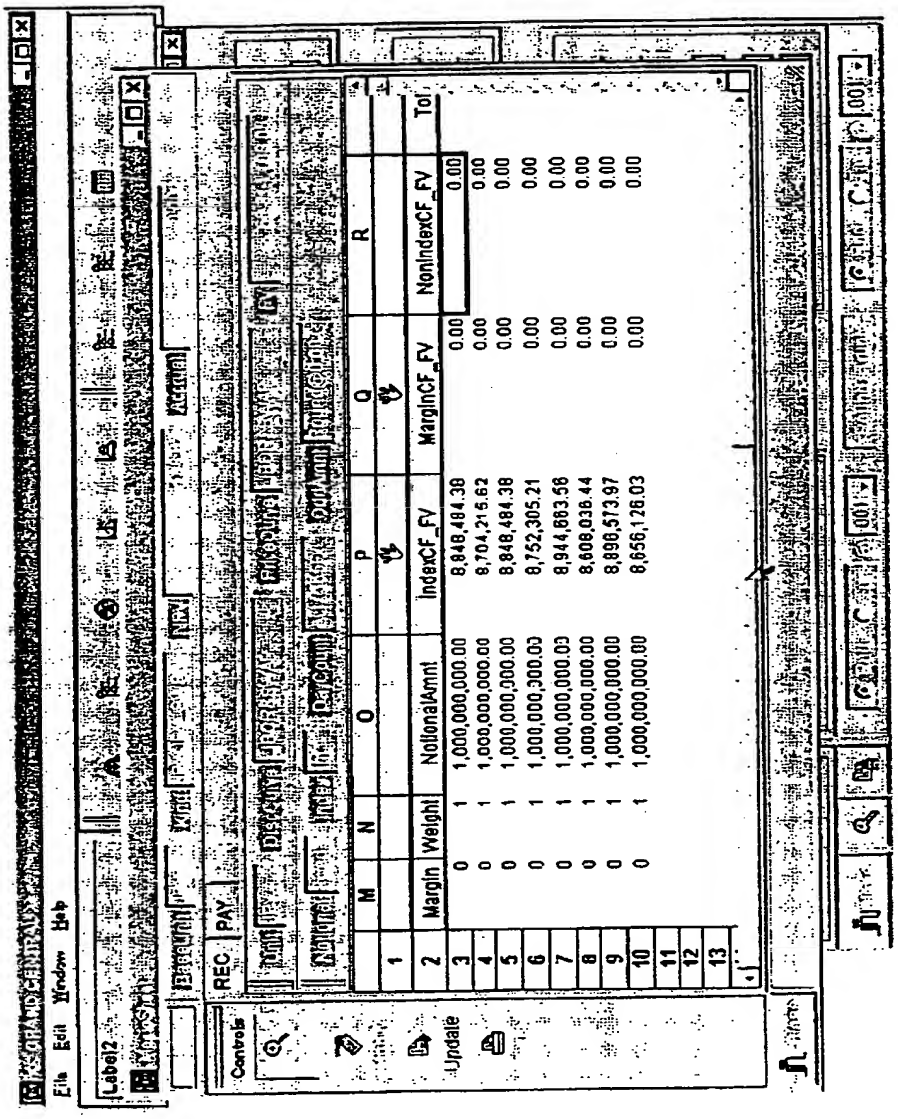
【図３２】

金利スワップ取引を作成するための画面例（その６）



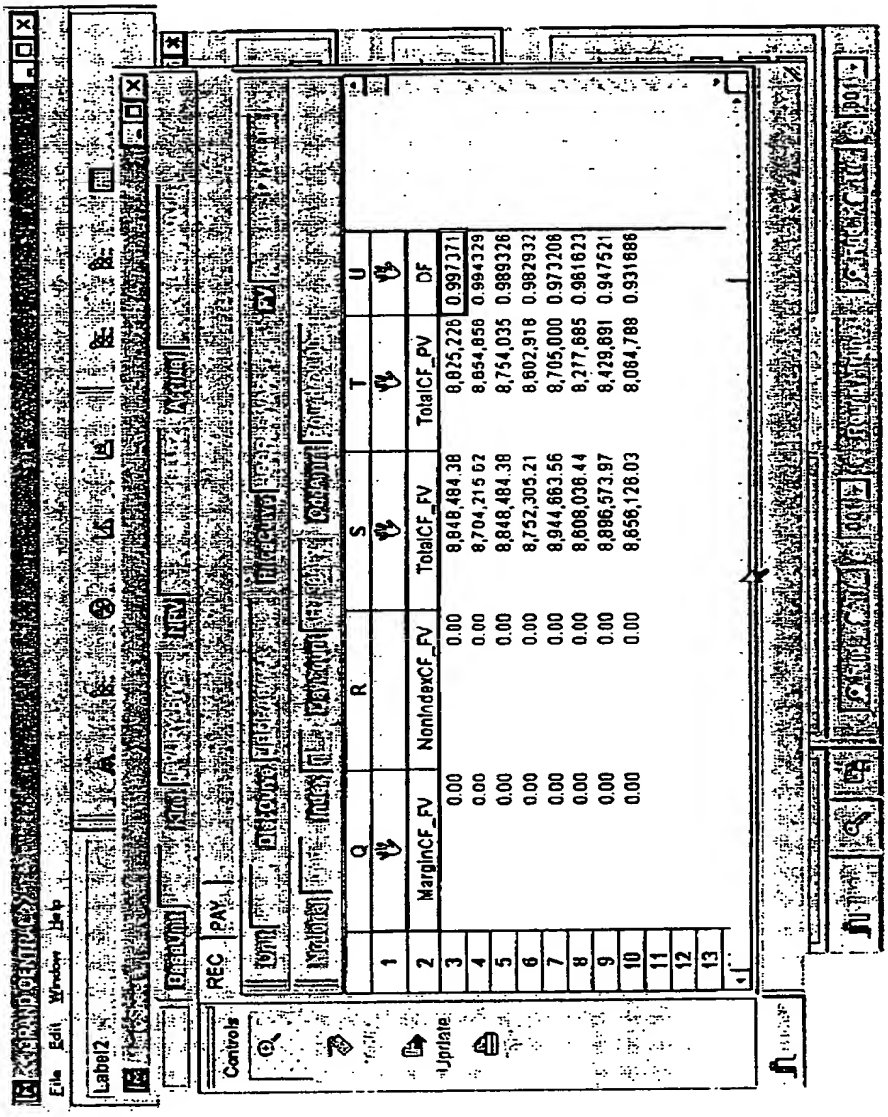
【図３３】

金利スワップ取引を作成するための画面例（その７）



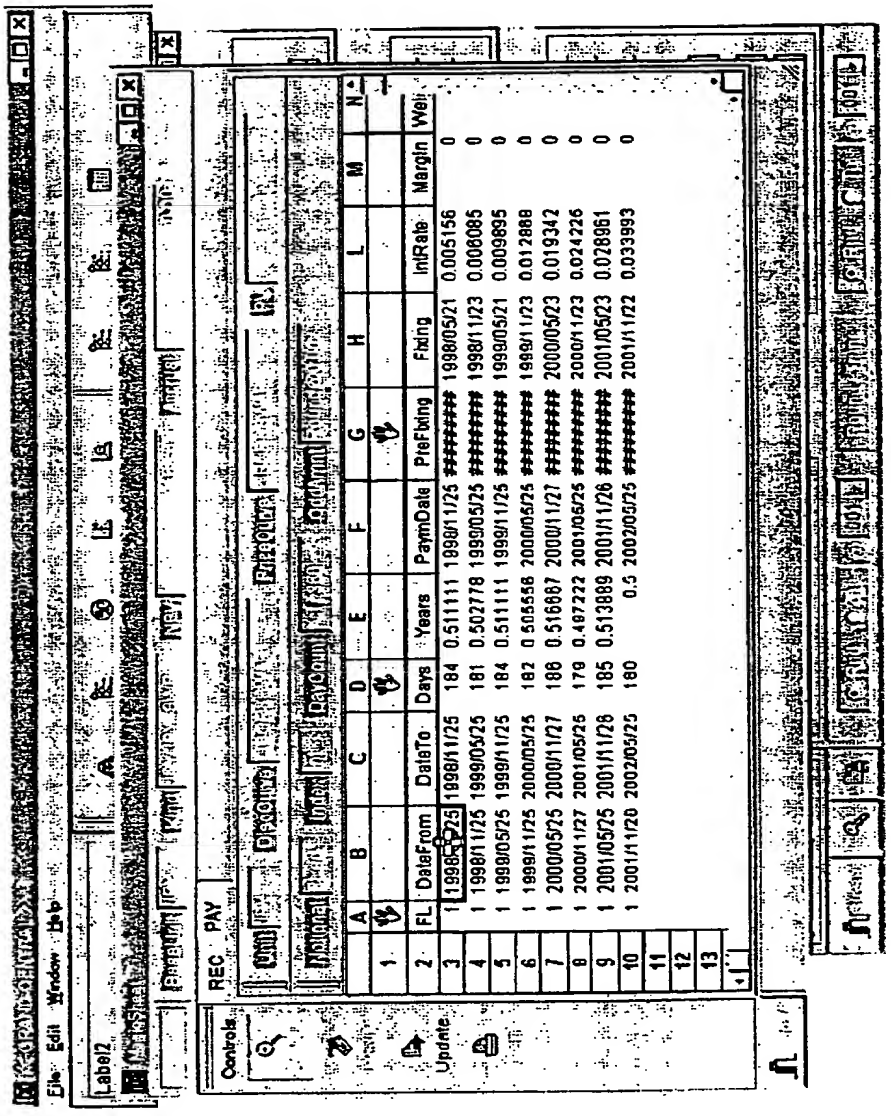
【図３４】

金利スワップ取引を作成するための画面例（その８）



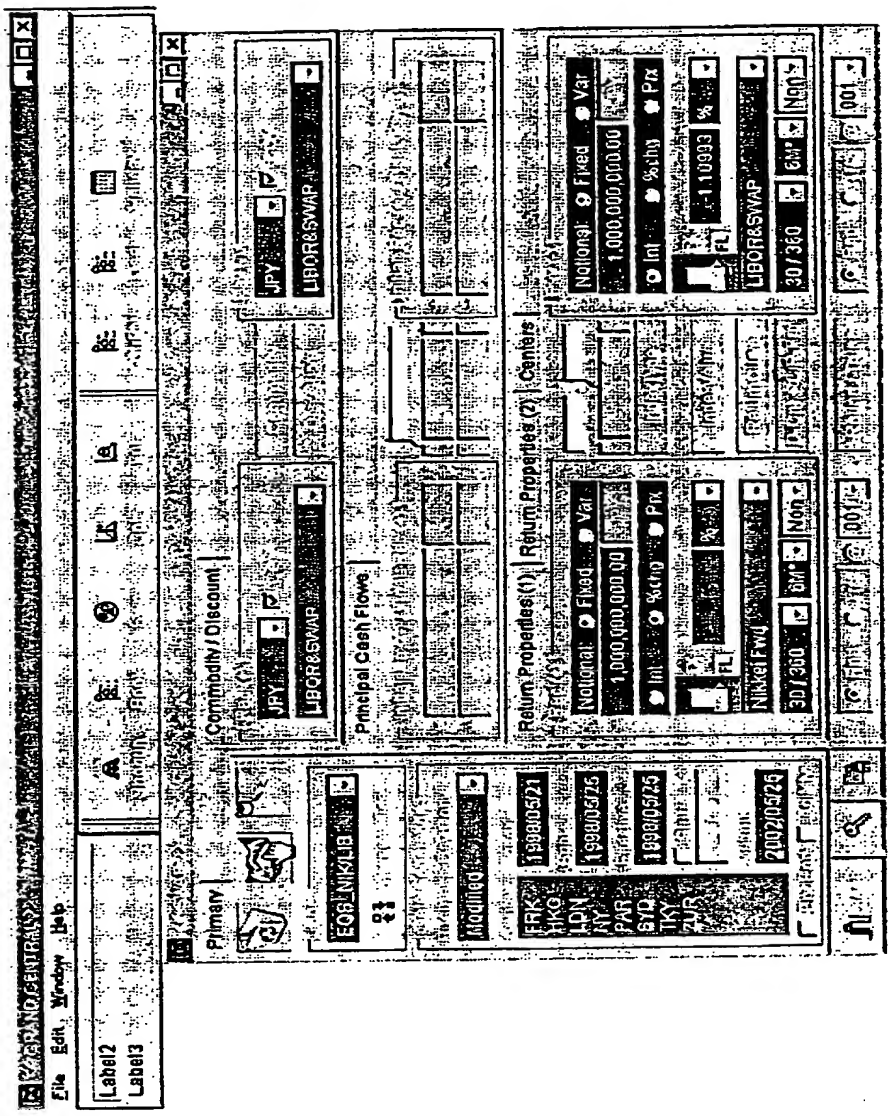
【図３５】

金利スワップ取引を作成するための画面例（その９）



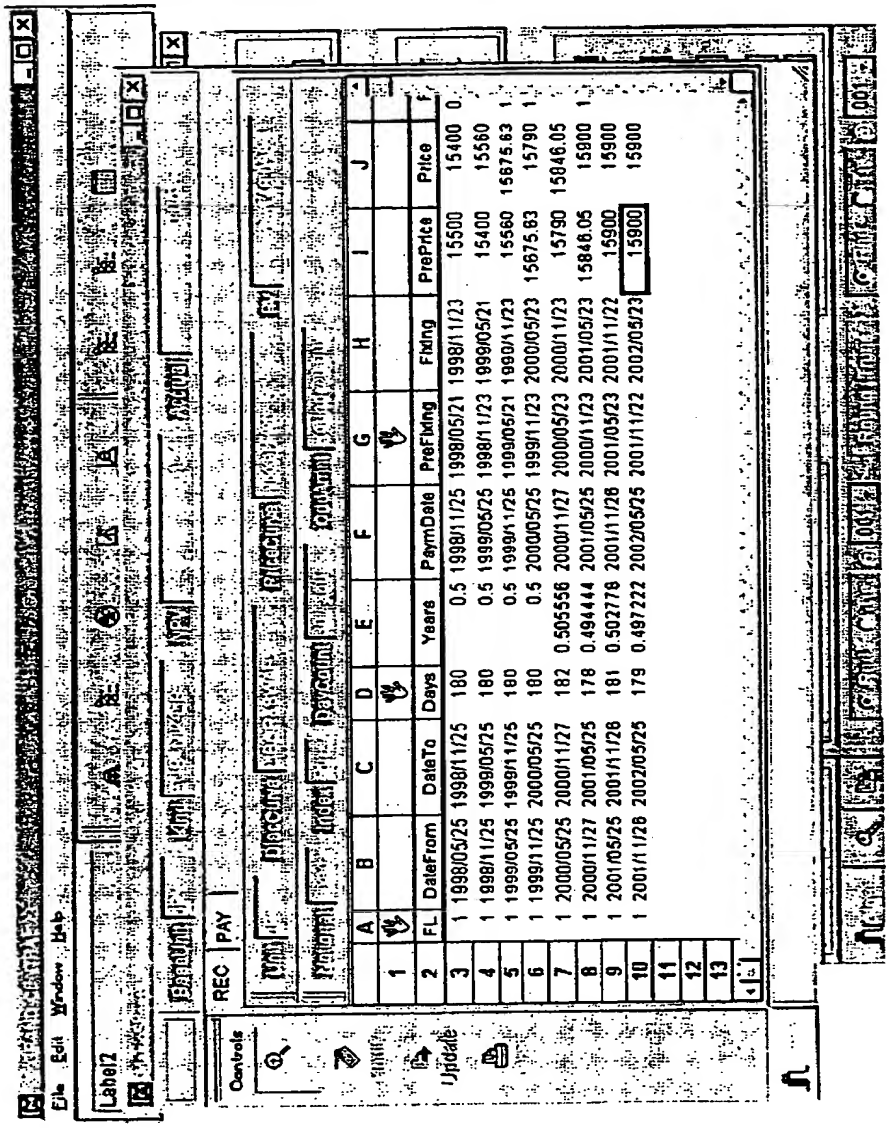
【図３６】

エクイティ・スワップ取引を作成するための画面例（その１）



【図３７】

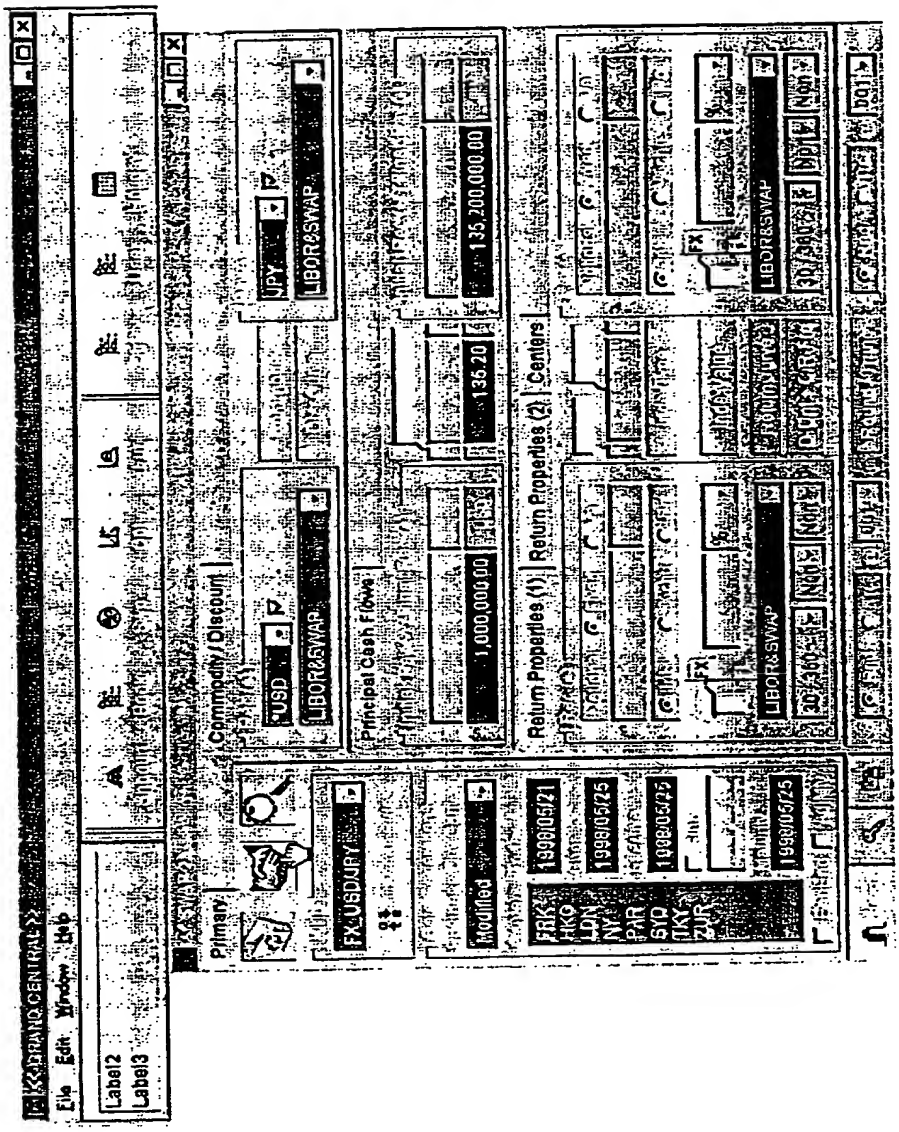
エクイティ・スワップ取引を作成するための画面例（その２）



	A	B	C	D	E	F	G	H	I	J	
1											
2	FL	DateFrom	DateTo	Days	Years	PaymDate	PreFixing	Fixing	PrePrice	Price	F
3	1	1998/05/25	1998/11/25	180	0.5	1998/11/25	1998/05/21	1998/11/23	15500	15400	0
4	1	1998/11/25	1999/05/25	180	0.5	1999/05/25	1998/11/23	1999/05/21	15400	15560	
5	1	1999/05/25	1999/11/25	180	0.5	1999/11/25	1999/05/21	1999/11/23	15560	15675.83	1
6	1	1999/11/25	2000/05/25	180	0.5	2000/05/25	1999/11/23	2000/05/23	15675.83	15790	1
7	1	2000/05/25	2000/11/27	182	0.505556	2000/11/27	2000/05/23	2000/11/23	15790	15846.05	
8	1	2000/11/27	2001/05/25	178	0.494444	2001/05/25	2000/11/23	2001/05/23	15846.05	15800	1
9	1	2001/05/25	2001/11/28	181	0.502778	2001/11/28	2001/05/23	2001/11/22	15900	15900	
10	1	2001/11/28	2002/05/25	179	0.497222	2002/05/25	2001/11/22	2002/05/23	15900	15900	
11											
12											
13											

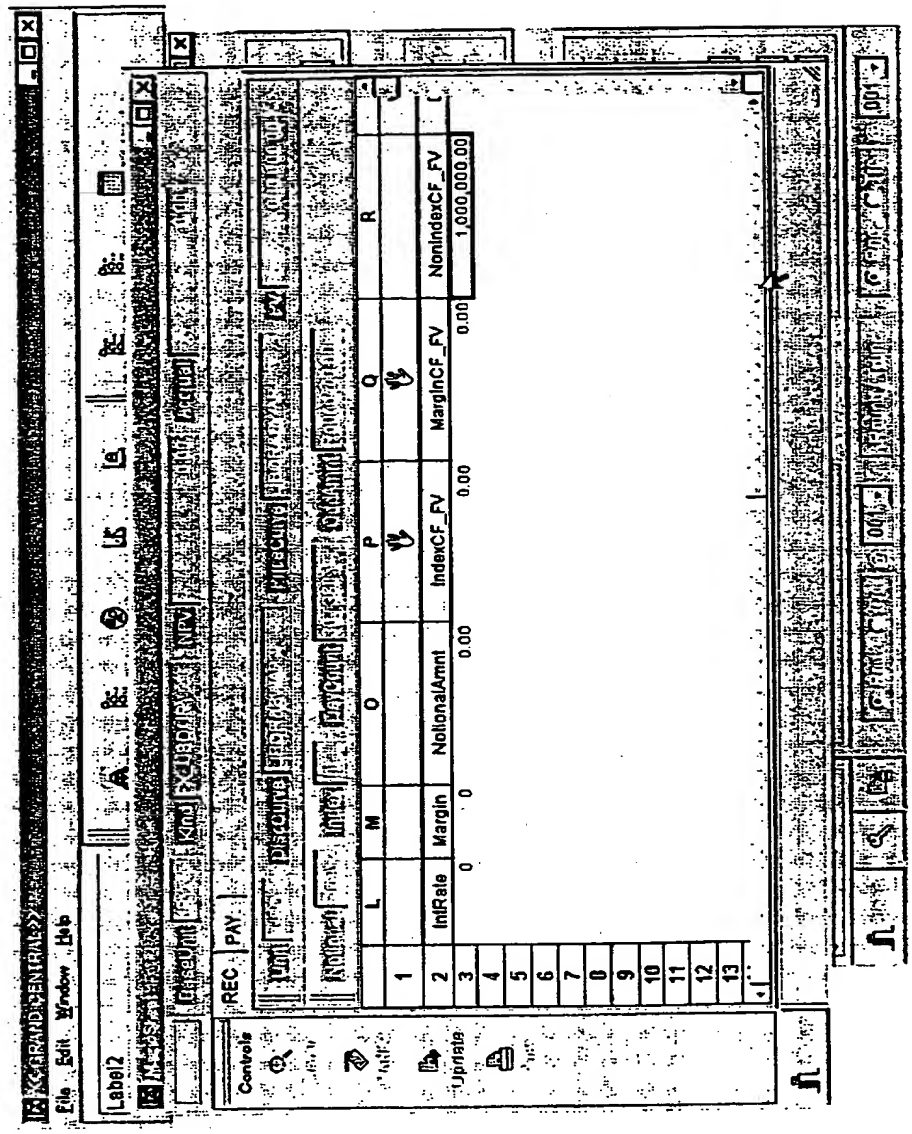
【図３８】

為替取引を作成するための画面例（その１）



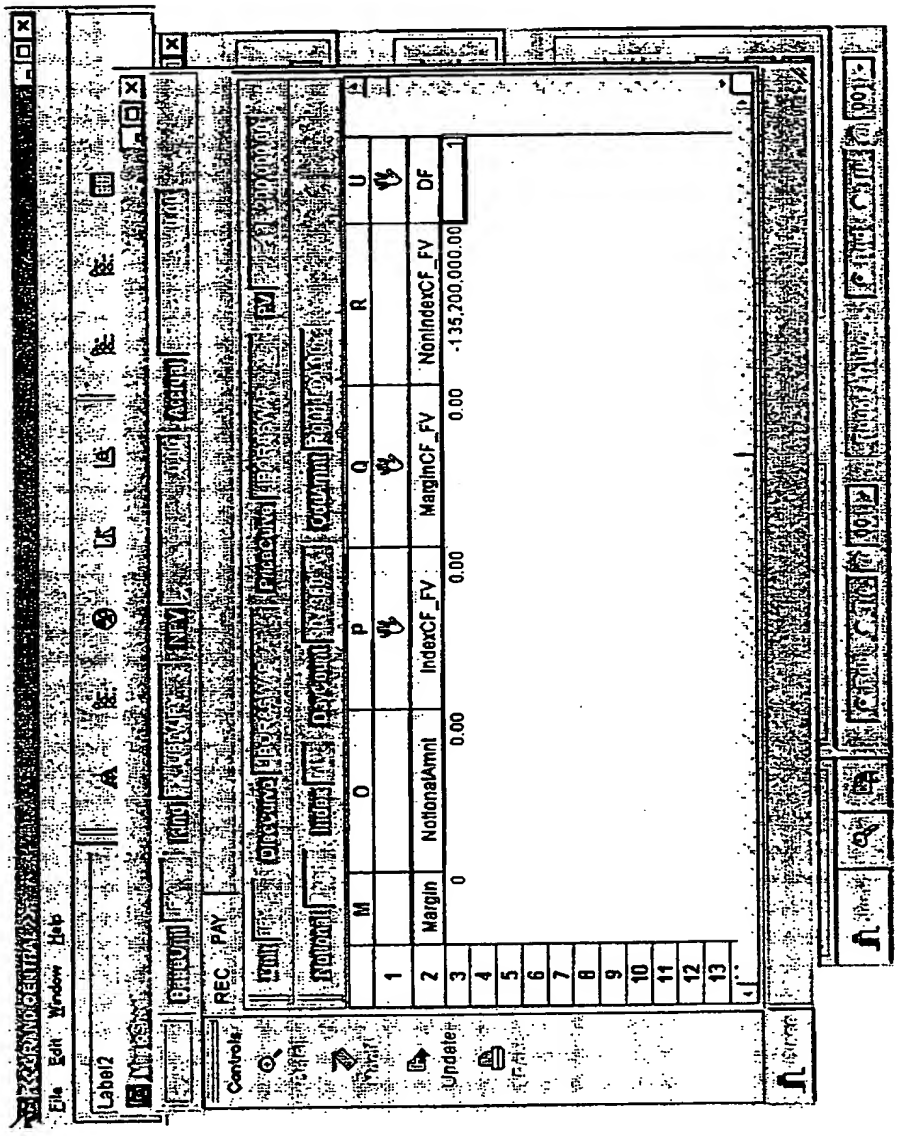
【図３９】

為替取引を作成するための画面例（その２）



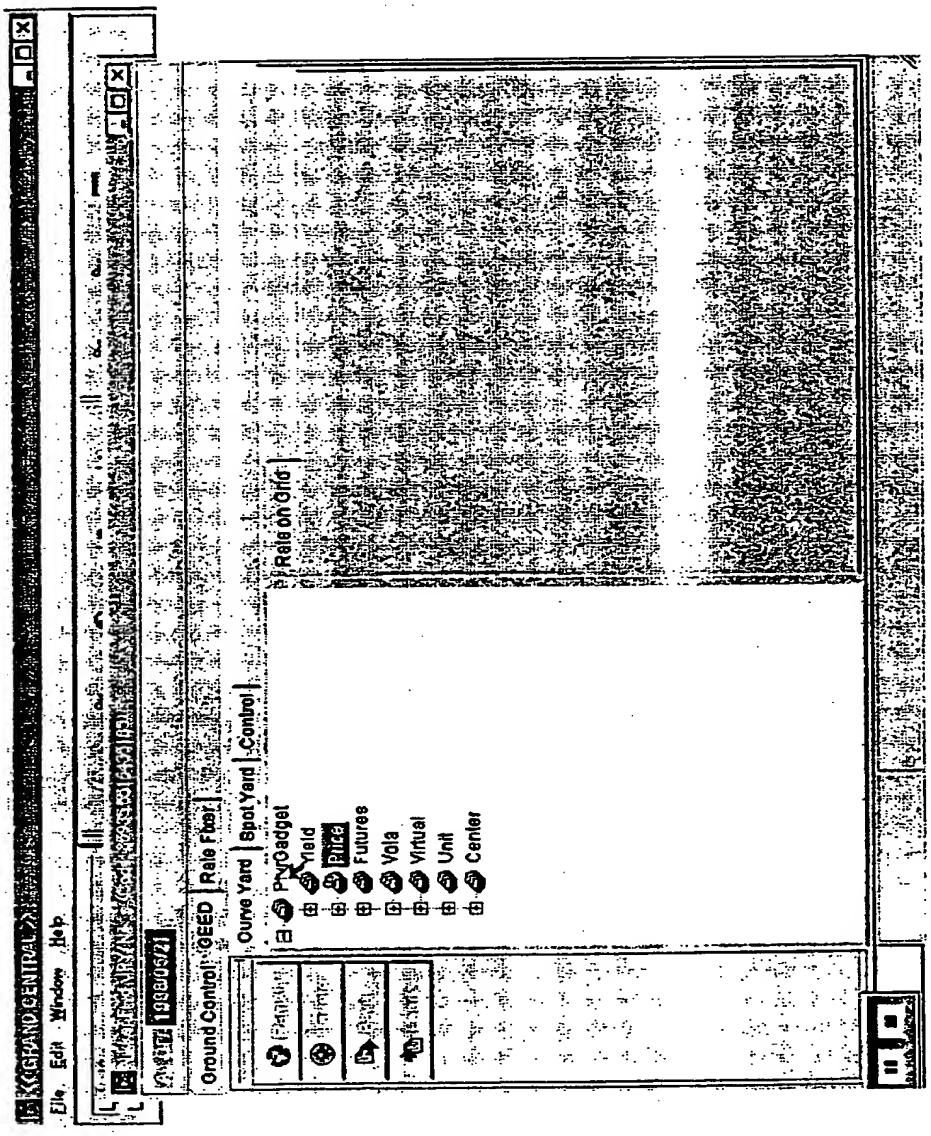
【図４０】

為替取引を作成するための画面例（その３）



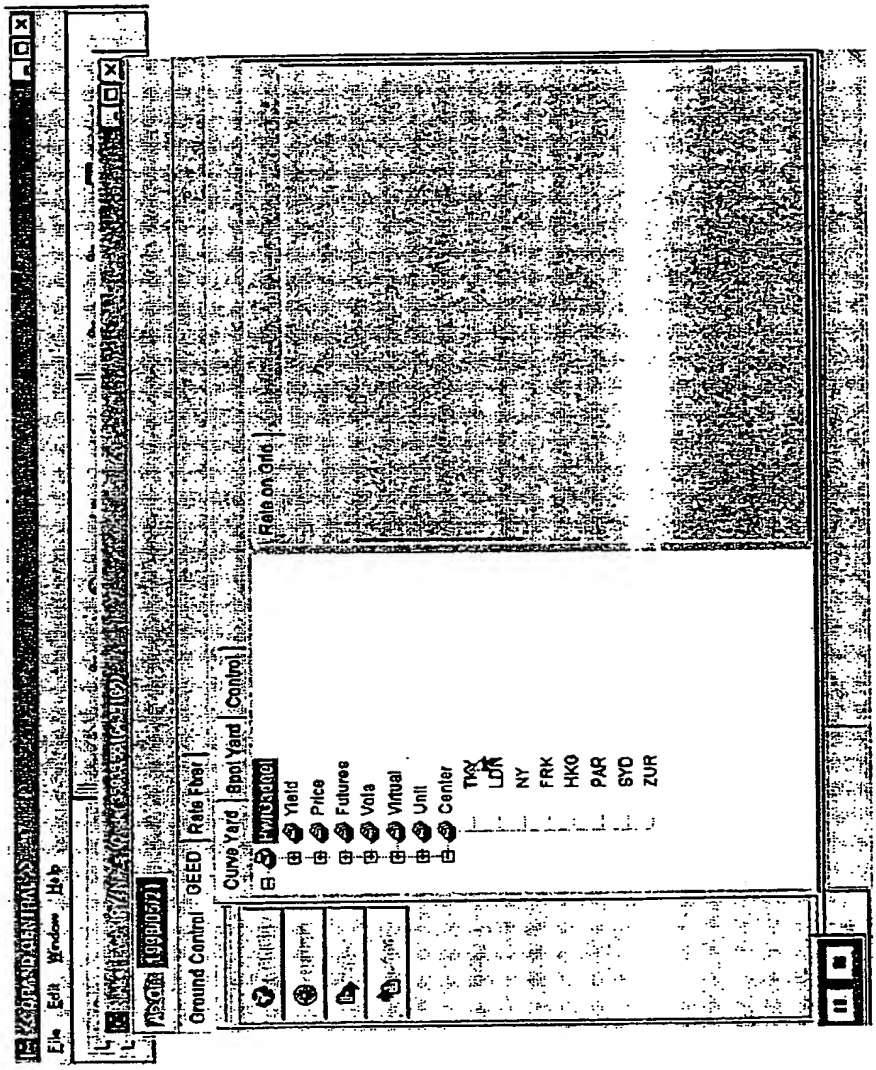
【図４１】

"PwrGadget" 機能の初期画面例



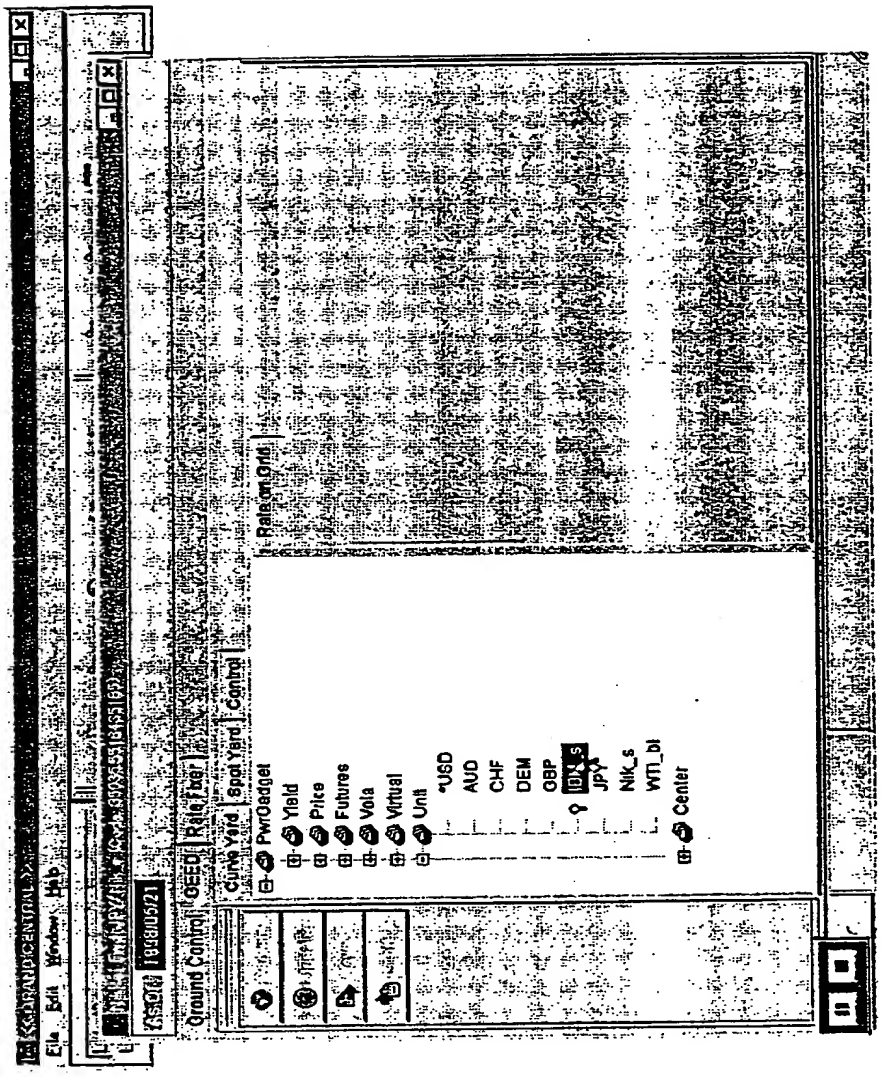
【図４２】

休日除外対象都市の定義画面例



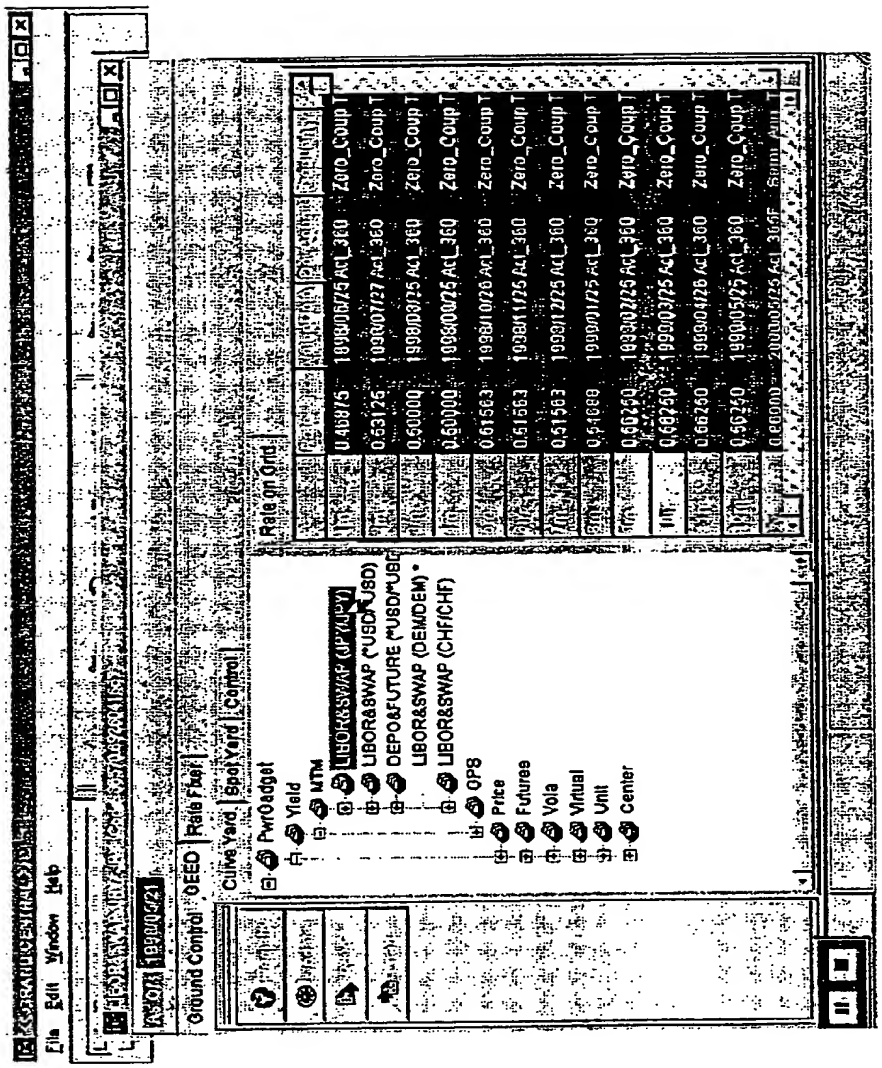
【図４３】

取引単位の定義画面例



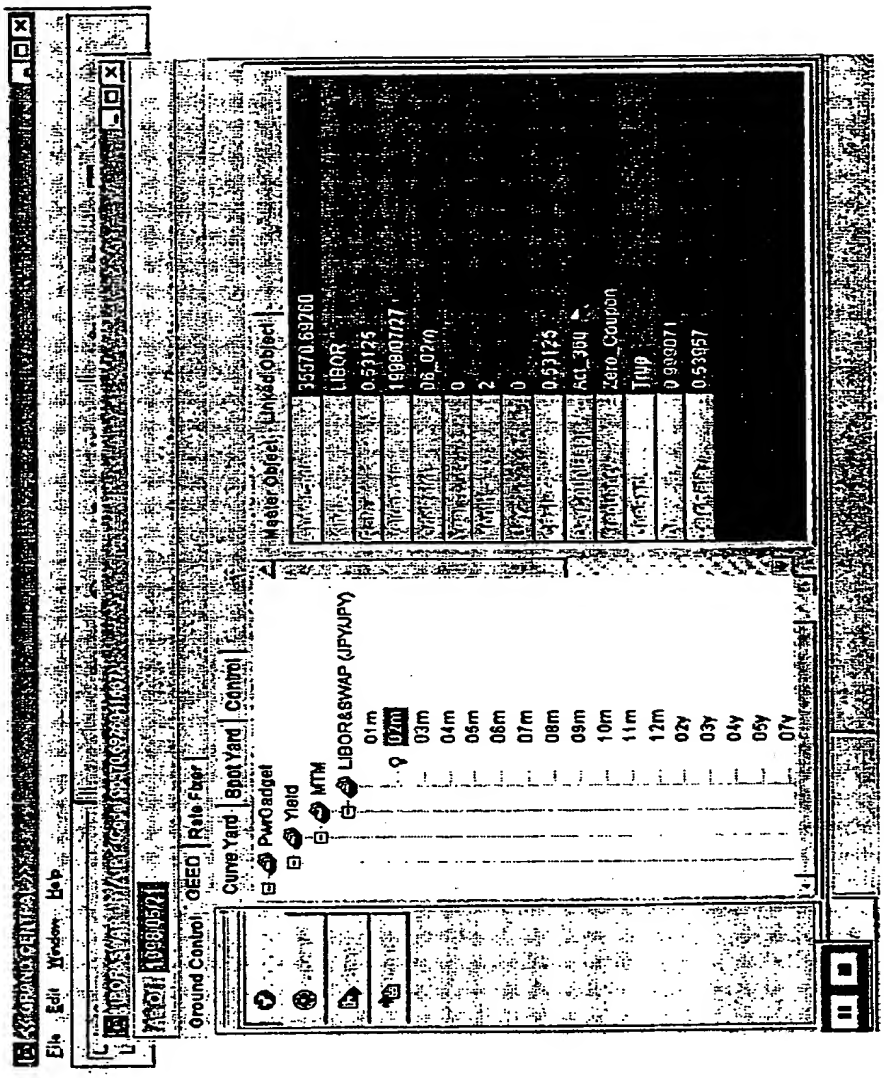
【図４４】

イールド・カーブの定義画面例（その１）



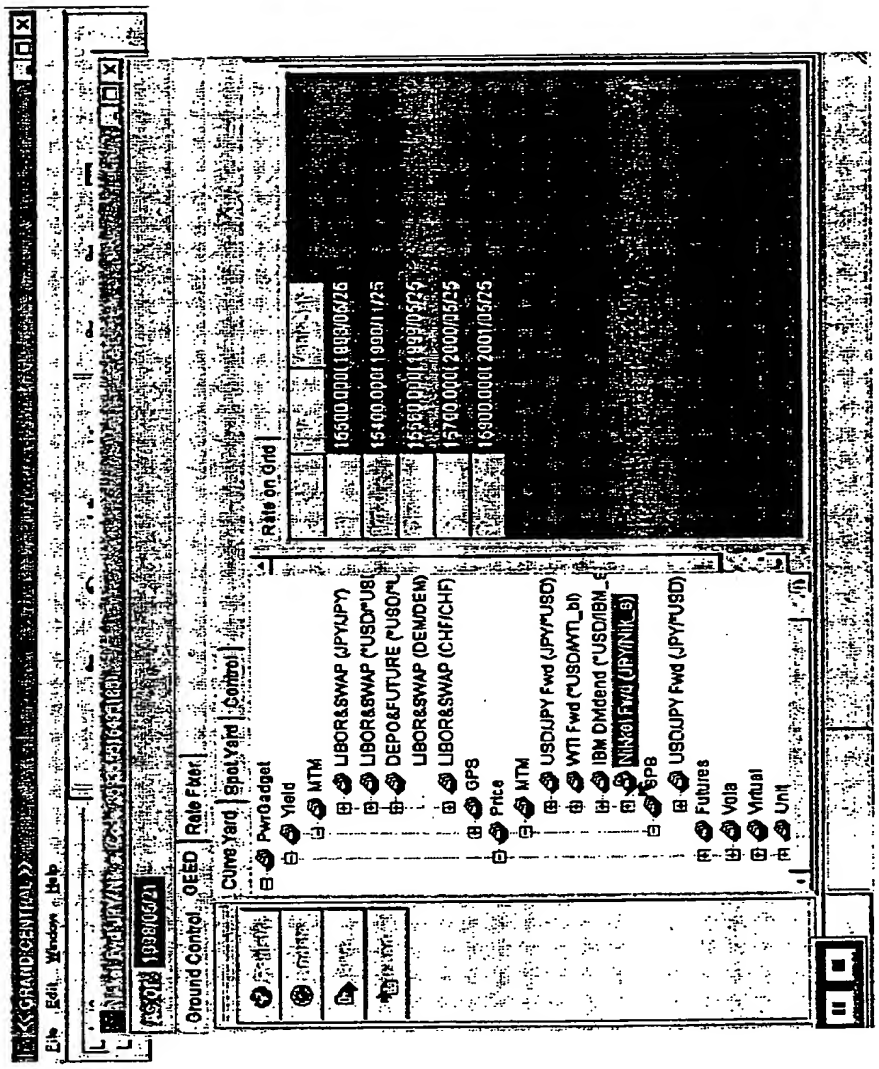
【図４５】

イールド・カーブの定義画面例（その２）



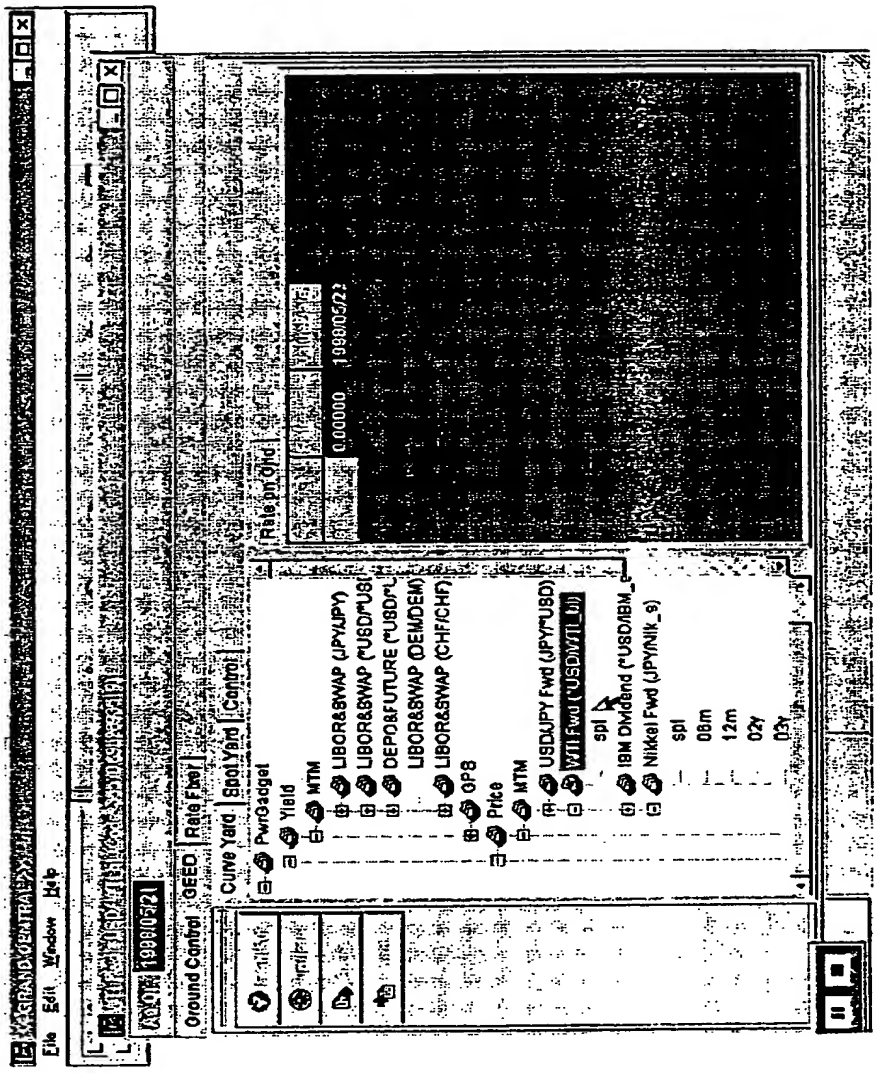
【図４６】

プライス・カーブの定義画面例（その１）



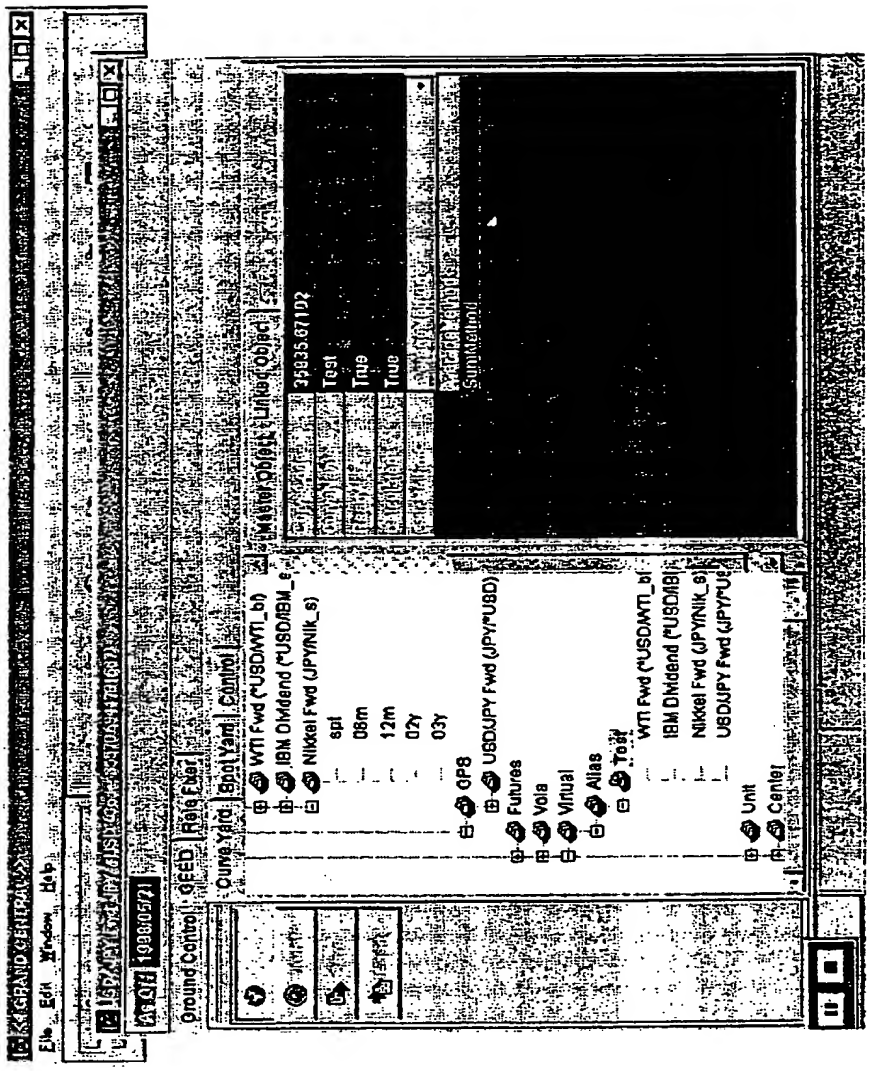
【図４７】

プライス・カーブの定義画面例（その２）



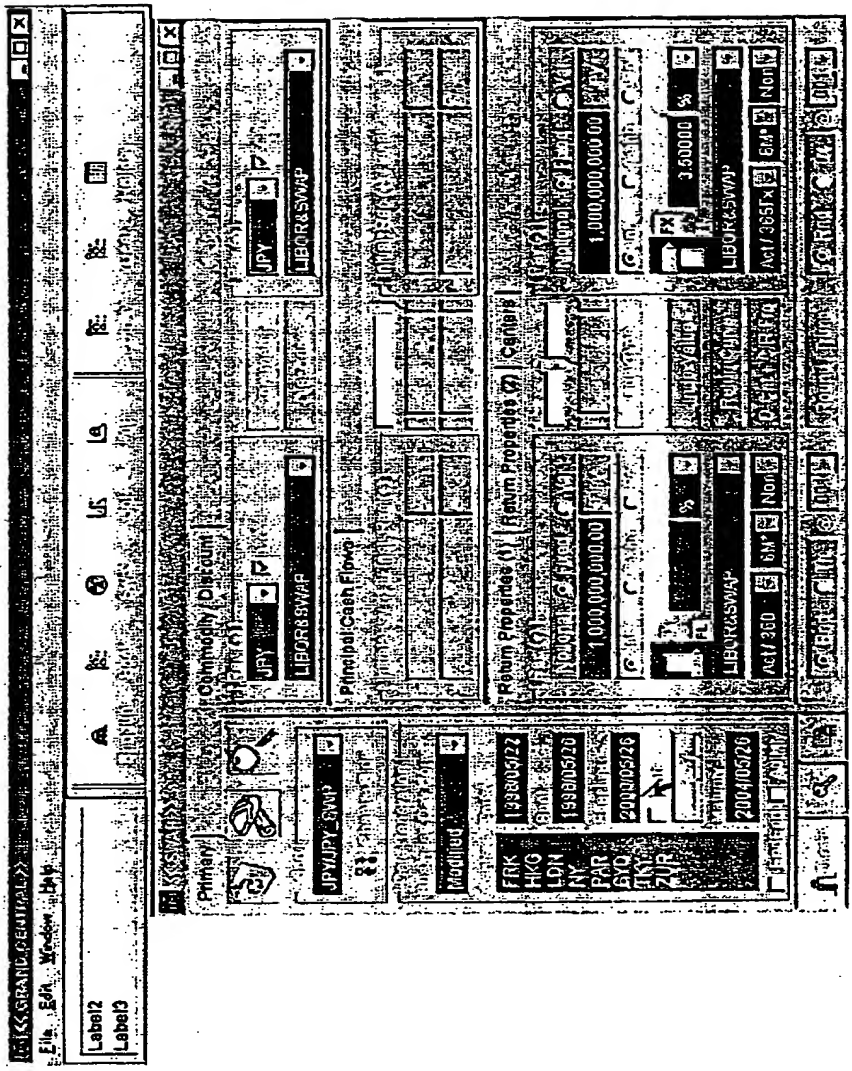
【図４８】

仮想カーブの定義画面例



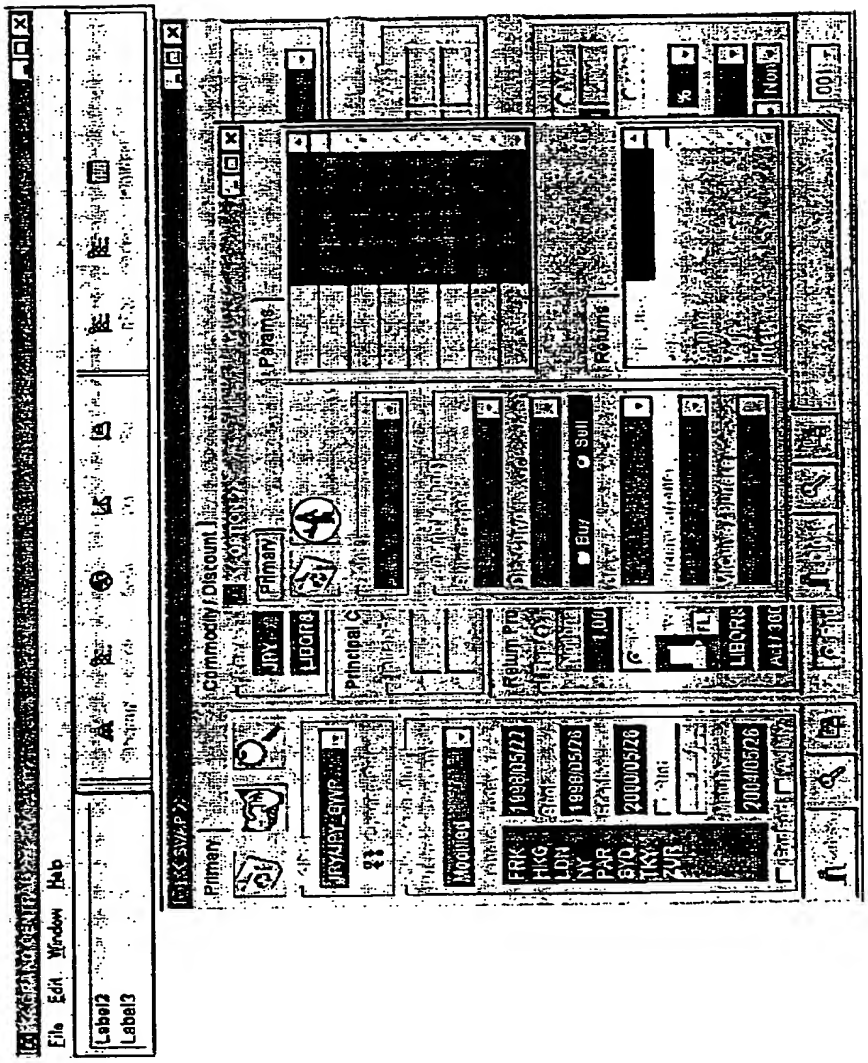
【図４９】

オプション取引において原資産である
スワップ取引を作成するための画面例



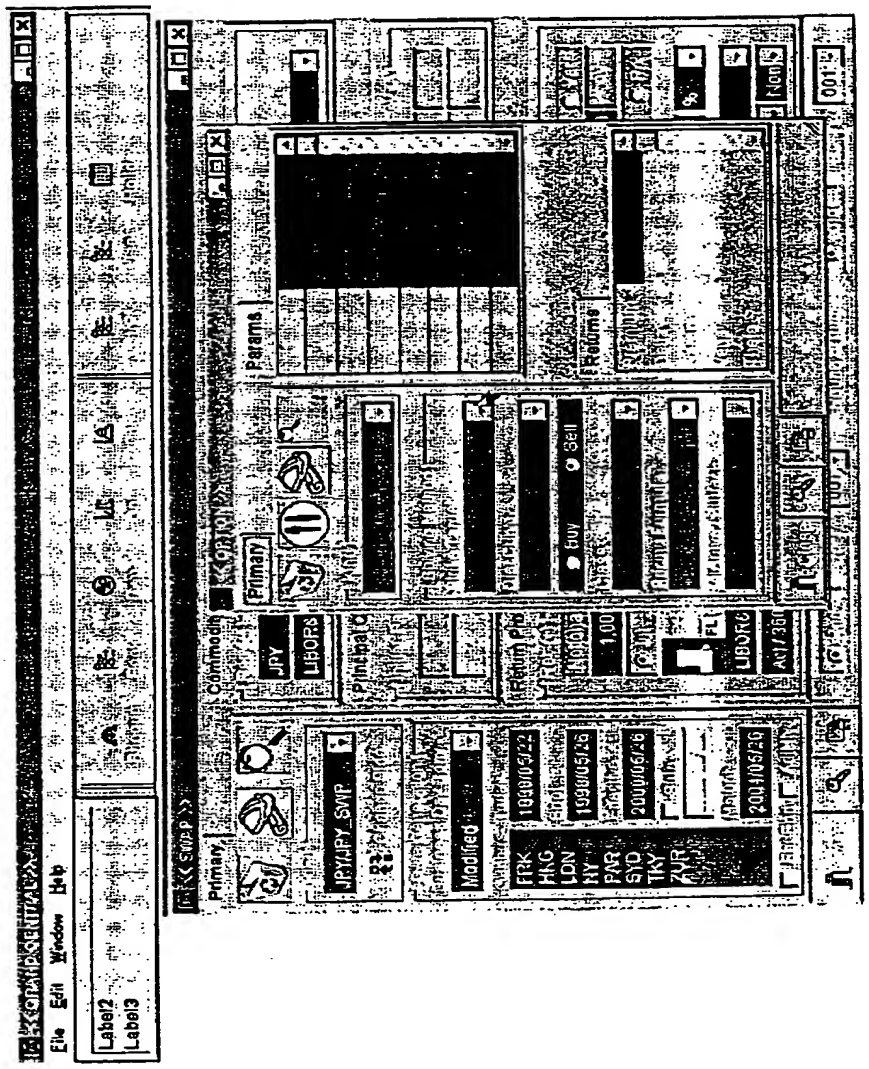
【図５０】

オプション取引の定義ウインドウの画面例



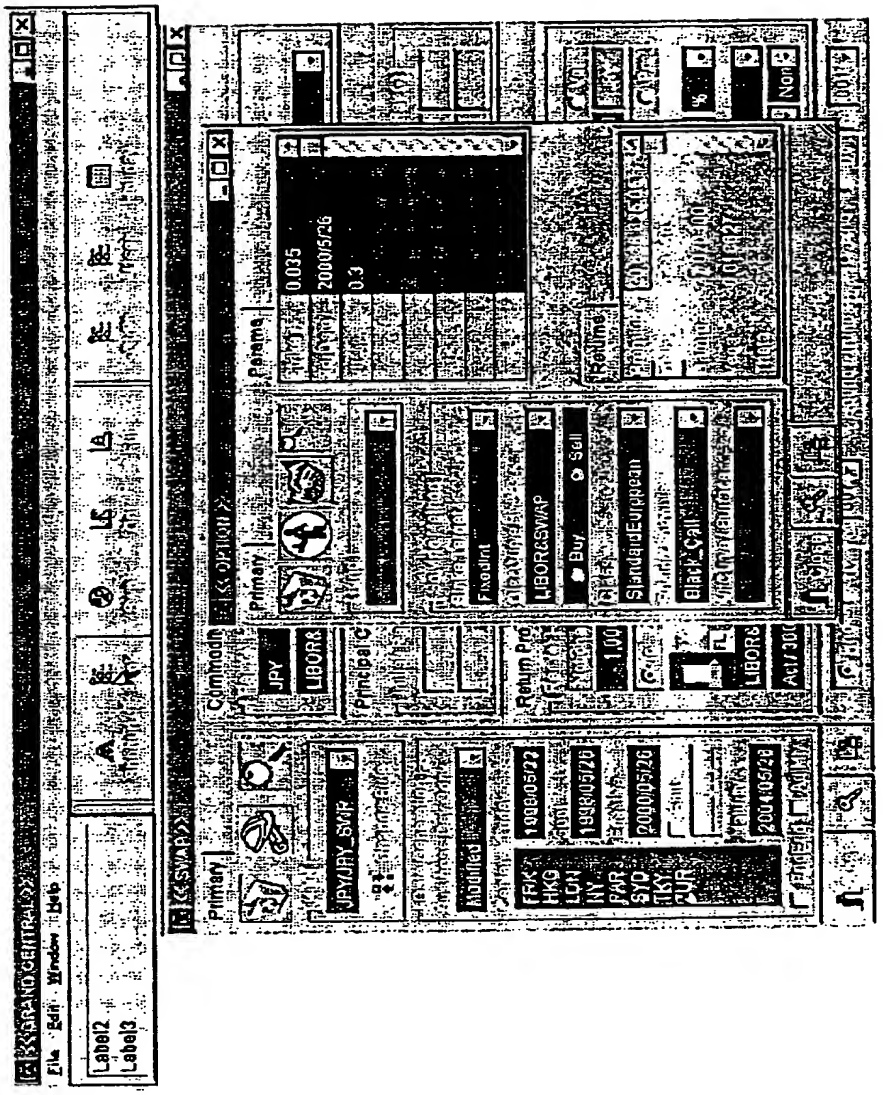
【図５１】

オプション取引を作成するための画面例（その１）



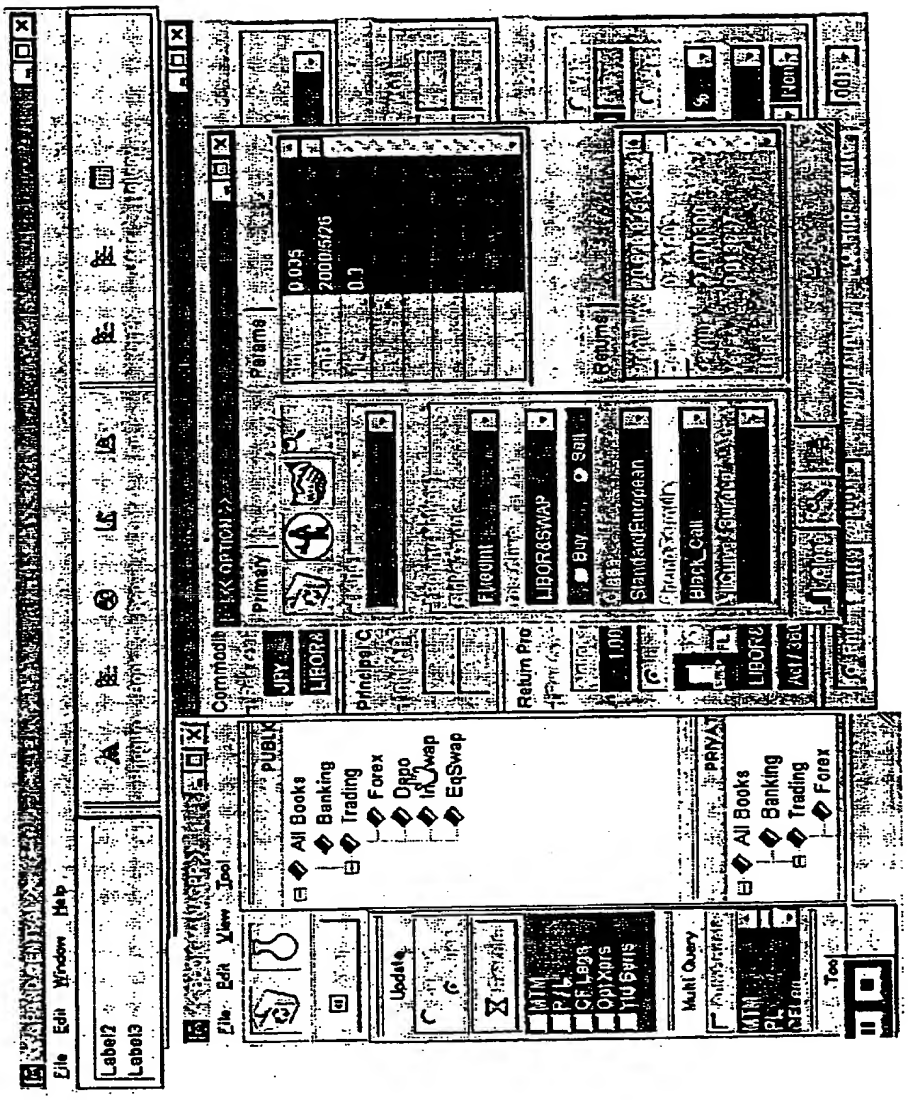
【図５２】

オプション取引を作成するための画面例（その２）



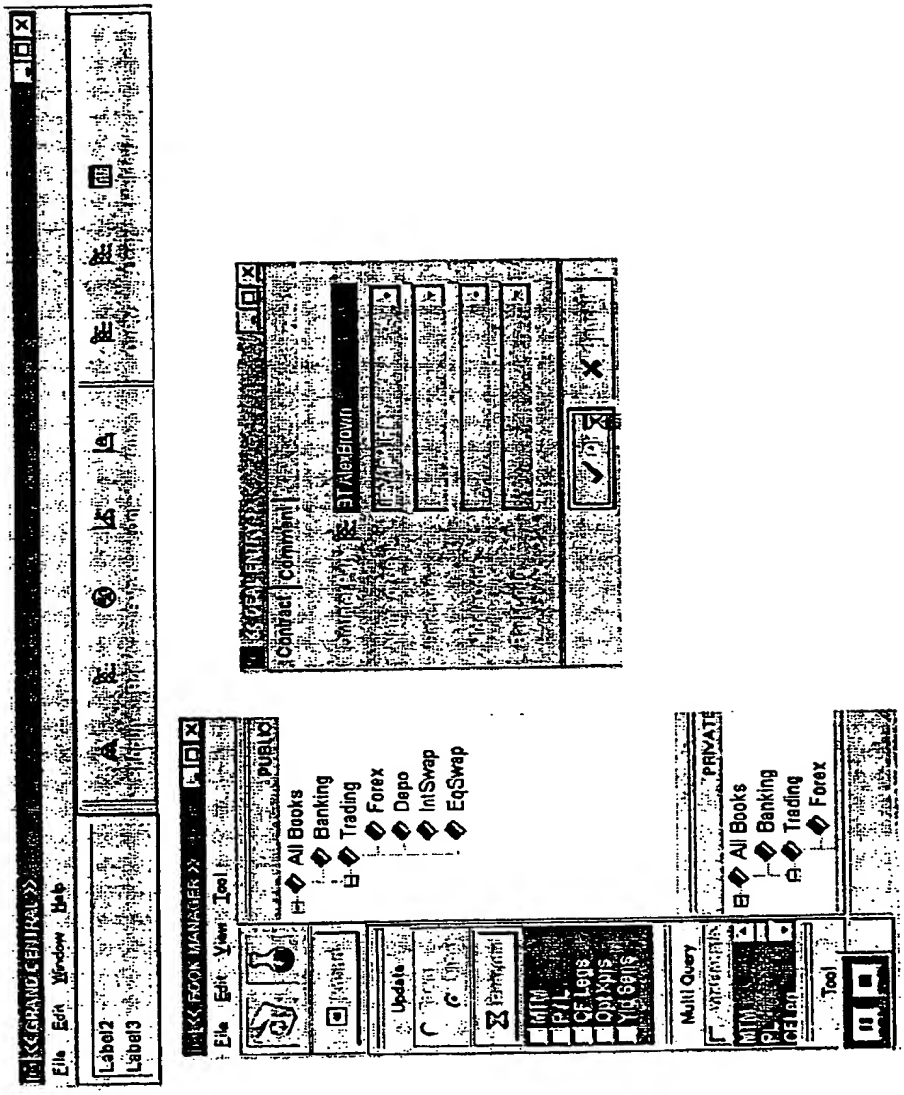
【図５３】

金融取引の合成手順の画面例（その１）



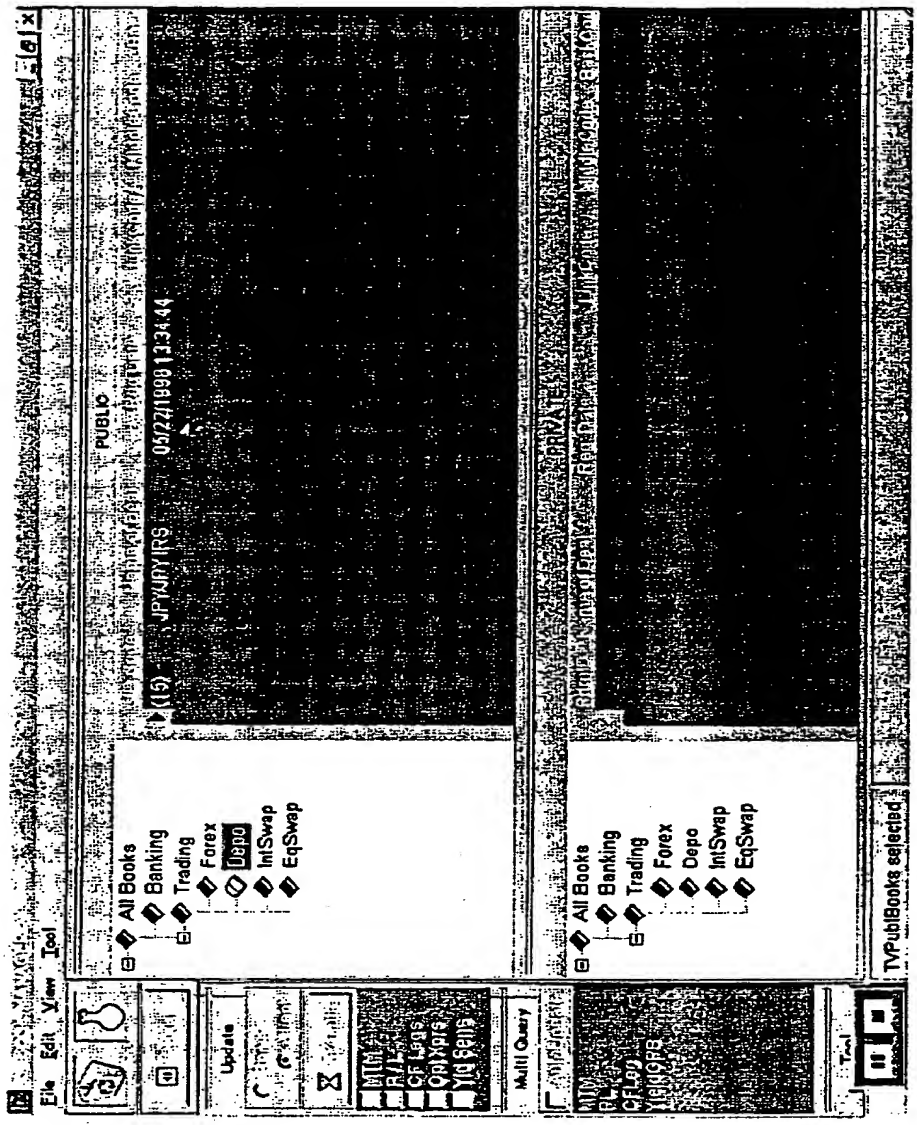
【図５４】

金融取引の合成手順の画面例（その２）



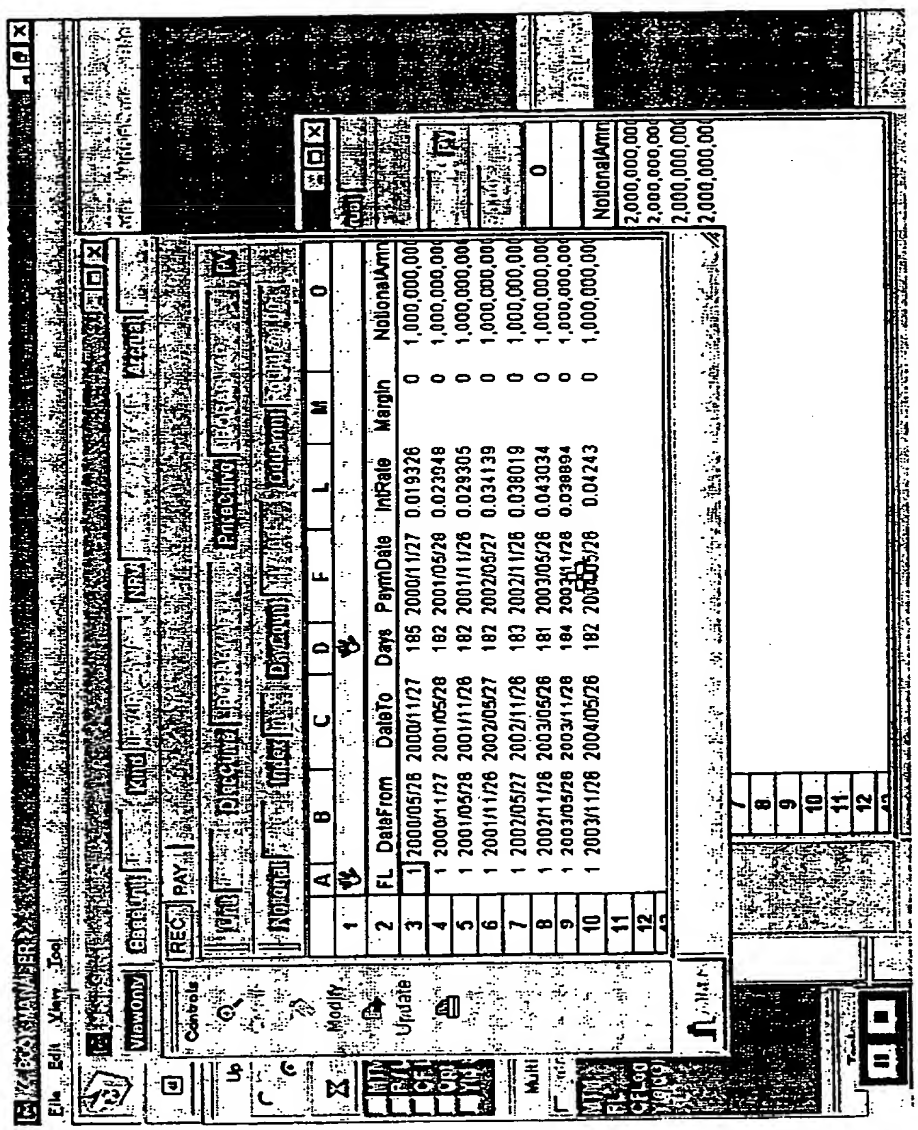
【図５５】

金融取引の合成手順の画面例（その３）



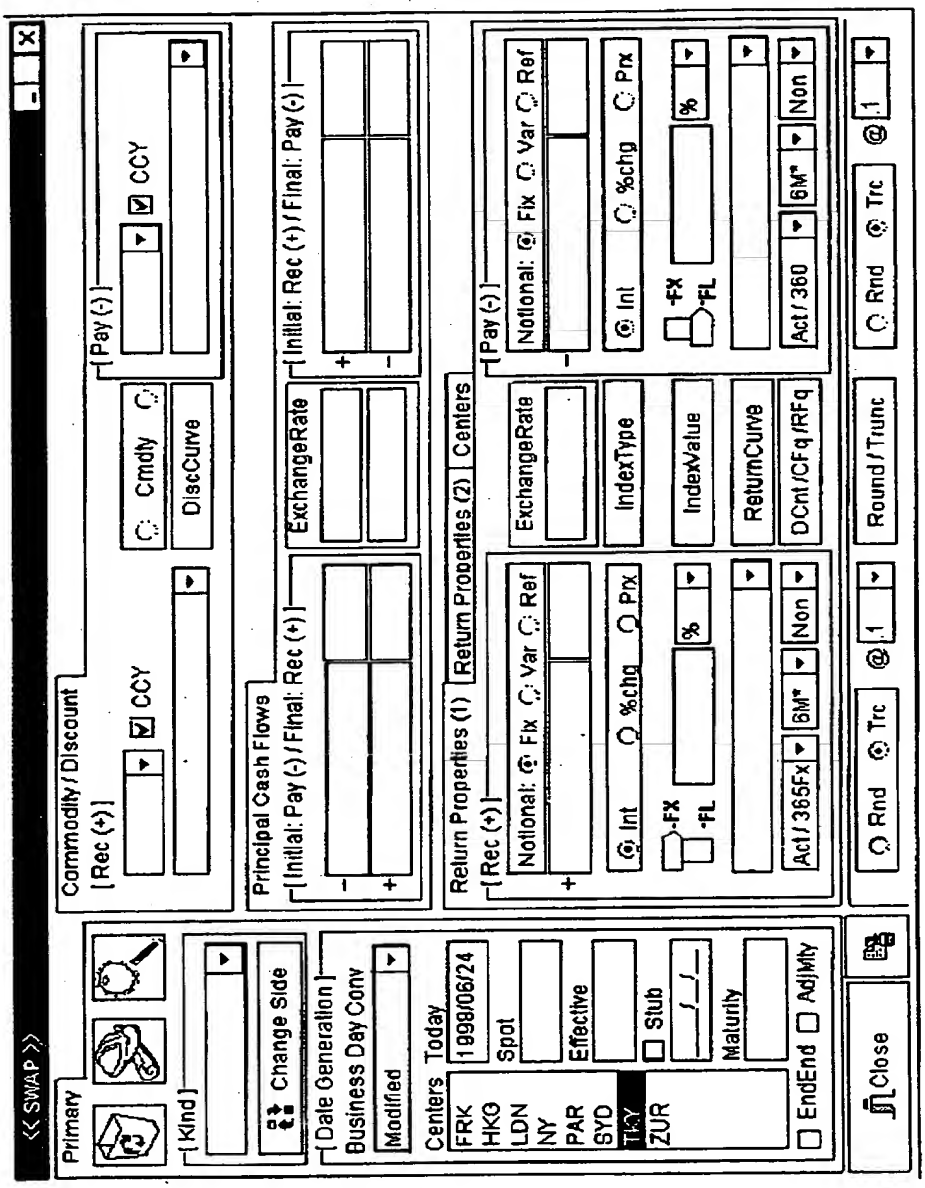
【図５６】

金融取引の合成手順の画面例（その４）



【図５７】

キャッシュ・フロー系取引を作成するための初期画面例（強調図）



【書類名】　　要約書

【要約】

【課題】　シンプルなシステム構成を実現することにより、金融取引に対する統合リスク管理システムの開発・運用コストを低減させ、システム性能を向上させることにある。

【解決手段】　複数の取引実体に対応するクラスをとりまとめて１つの仮想取引が実現される。キャッシュ・フロー系取引実体が、受け側・払い側のそれぞれにおける単位取引期間ごとのキャッシュ・フロー要素（CashFlowLet ）の集合として管理され、各要素ごとの時価評価演算が共通化される。オプション取引が、原資産取引のクラスをコンテナとして格納するクラスによって実現される。金融カーブ定義機能と、複数の金融カーブを合成して１つの仮想カーブを実現する機能とが実装される。リスク管理のためのパラメータ変更とそれに対するシミュレーション結果の表示を容易に行えるユーザ・インタフェースが提供される。

【選択図】　　図１

出証特平１１－３０５１６５５

【書類名】　職権訂正データ
【訂正書類】　特許願

＜認定情報・付加情報＞
【特許出願人】
【識別番号】　598086327
【住所又は居所】　アメリカ合衆国　ニューヨーク州　ニューヨーク市　リバティー・ストリート　１３０
【氏名又は名称】　バンカース・トラスト・カンパニー（バンカース・トラスト銀行）　◇◇
【代理人】　申請人
【識別番号】　100106851
【住所又は居所】　東京都千代田区二番町８番地の２０　二番町ビル
【氏名又は名称】　野村　泰久
【選任した代理人】
【識別番号】　100102336
【住所又は居所】　神奈川県海老名市中央一丁目１８番２７号　士業ビル３階　澁谷・久保田特許事務所
【氏名又は名称】　久保田　直樹

【書類名】　出願人名義変更届

【整理番号】　BT98001

【提出日】　平成11年 6月25日

【あて先】　特許庁長官殿

【事件の表示】

【出願番号】　平成10年特許願第183133号

【承継人】

【識別番号】　399033784

【住所又は居所】　東京都文京区小石川１－４－１

【氏名又は名称】　アイキュー・ファイナンシャル・システムズ・ジャパン株式会社

【承継人代理人】

【識別番号】　100106851

【弁理士】

【氏名又は名称】　野村　泰久

【電話番号】　03-3238-0158

【手数料の表示】

【予納台帳番号】　041391

【納付金額】　4,600円

【提出物件の目録】

【包括委任状番号】　9906370

【書類名】　手続補正書
【整理番号】　BT98001
【提出日】　平成11年 6月30日
【あて先】　特許庁長官殿
【事件の表示】
　【出願番号】　平成10年特許願第183133号
【補正をする者】
　【識別番号】　399033784
　【氏名又は名称】　アイキュー・ファイナンシャル・システムズ・ジャパン株式会社
【代理人】
　【識別番号】　100106851
　【弁理士】
　【氏名又は名称】　野村　泰久
　【電話番号】　03-3238-0158
【手続補正　1】
　【補正対象書類名】　特許願
　【補正対象項目名】　発明者
　【補正方法】　変更
　【補正の内容】
　　【発明者】
　　【住所又は居所】　東京都渋谷区鶯谷町１６－４
　　【氏名】　山崎　裕
　　【発明者】
　　【住所又は居所】　千葉県船橋市中野木１－１８－２５　グリーンビレッジ東船橋２０２号
　　【氏名】　中西　芳明
　　【発明者】
　　【住所又は居所】　神奈川県横浜市青葉区荏田西５－１－２８

【氏名】　薄葉　真哉

【発明者】

【住所又は居所】　神奈川県横浜市港北区師岡町２６２－３－２－１０３

【氏名】　山下　司

【発明者】

【住所又は居所】　東京都港区芝浦４－１８－３０

【氏名】　角南　秀幸

【その他】　今回、本願を基礎とする優先権主張にて外国出願をするにあたり、発明者山下司の住所が変更になったことを願書に反映する必要があるため補正します。

認定・付加情報

特許出願の番号	平成１０年　特許願　第１８３１３３号
受付番号	５９９００６３０７１２
書類名	手続補正書
担当官	濱谷　よし子　　１６１４
作成日	平成１１年　８月２０日

＜認定情報・付加情報＞

【補正をする者】

【識別番号】　399033784

【住所又は居所】　東京都文京区小石川１－４－１

【氏名又は名称】　アイキュー・ファイナンシャル・システムズ・ジャパン株式会社

【代理人】　申請人

【識別番号】　100106851

【住所又は居所】　東京都千代田区二番町８番地の２０　二番町ビル

【氏名又は名称】　野村　泰久

次頁無

出　願　人　履　歴　情　報

識別番号　　〔５９８０８６３２７〕

１．変更年月日　　１９９８年　６月３０日
［変更理由］　新規登録
住　所　アメリカ合衆国　ニューヨーク州　ニューヨーク市　リバティー・ストリート　１３０
氏　名　バンカース・トラスト・カンパニー（バンカース・トラスト銀行）　◇◇

２．変更年月日　　１９９８年１２月　７日
［変更理由］　名称変更
住　所　アメリカ合衆国　ニューヨーク州　ニューヨーク市　リバティー・ストリート　１３０
氏　名　バンカース・トラスト・カンパニー

出　願　人　履　歴　情　報

識別番号　　〔３９９０３３７８４〕

１．変更年月日　　１９９９年　６月　１日
［変更理由］　新規登録
住　所　東京都文京区小石川１－４－１
氏　名　アイキュー・ファイナンシャル・システムズ・ジャパン株式会社